The Dissertation Committee for Cheryl Lynn Brabec Certifies that this is the approved version of the following Dissertation:

Directional Neutron Surveys on a Semiautonomous Mobile Robot for a Radiological Vault

Committee:

Sheldon Landsberger, Supervisor

Rajendra Vaidya, Co-Supervisor

William Charlton

Derek Haas

Drew Kornreich

Mitchell Pryor

Directional Neutron Surveys on a Semiautonomous Mobile Robot for a Radiological Vault

by

Cheryl Lynn Brabec

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 2020

In a properly automated and educated world, then, machines may prove to be the true humanizing influence. It may be that machines will do the work that makes life possible and that human beings will do all the other things that make life pleasant and worthwhile.

—Isaac Asimov, *Robot Visions*

Acknowledgements

I would like to sincerely thank all the members of my committee: Bill Charlton, Derek Haas, Drew Kornreich, Sheldon Landsberger, Mitch Pryor, and Raj Vaidya. Special thank you to Raj for his mentoring starting in my early years at LANL. I will forever appreciate Drew's advice and for entertaining random questions I have (and making me learn FORTRAN). Last but not least, thank you to Sheldon for his guidance and kindness over the years. His support has been invaluable to me.

I have been fortunate to receive help from many others in the course of this work. I appreciate the initial guidance given by Steven Biegalski when this process started. Huge thank you to Ashlee Vrana for all her administrative aid. Thank you to Matthew Durbin for his assistance in performing the initial detector spatial experiments. I am indebted to many colleagues at LANL for their support and technical assistance including Cristy Abeyta, Jeff Archuleta, Pam Dominguez, Larry Duran, Alexander Feldman, David Grow, Stephen Jimenez, Daniel Leyba, Vicki Longmire, Leonard Manzanares, Anthony Nettleton, David Prochnow, Debbie Roybal, and Tresa Yarbro. Thank you to the offices of NA-19 and NA-53 for their financial assistance. Thank you to Gail Reitenbach for her exceptional editing abilities (and any mirstakes still present are wholly my own).

Thank you to my family and friends for your love and support. Finally, thank you to my dog who has been with me throughout this graduate school experience, and who reminds everyone to find joy in their lives.

Abstract

Directional Neutron Surveys on a Semiautonomous Mobile Robot for a Radiological Vault

Cheryl Lynn Brabec, Ph.D.

The University of Texas at Austin, 2020

Supervisor: Sheldon Landsberger

Co-Supervisor: Rajendra Vaidya

Monitoring of stored nuclear materials is time-consuming and can expose human workers to high levels of radiological risk. The use of robotics in the nuclear complex has the ability to provide increased monitoring while lowering worker dose. Radiation detectors are essential tools for assessing the hazards present in everyday work and can be incorporated with robotic systems to assist in their use. However, because there are often many materials collocated in storage areas, the ability to determine the incident radiation direction—and thus the source—of radiation in a crowded room would provide a better understanding of where the radiological hazards are. A Bridgeport Instruments boron-10 neutron scintillator detector was well characterized to determine inherent directional dependencies. Modeling of a neutron shield to create an artificial aperture around the neutron detector was performed in Monte Carlo N-Particle Transport Code. Once the shielding was fabricated, the performance of the aperture was evaluated with a Cf-252 neutron source. The shielding resulted in a maximum 21.75% difference in radiation coming directly through the aperture versus from a different direction. A Pioneer LX robot

was successfully approved for operation in the main plutonium facility at Los Alamos National Laboratory (LANL). The neutron detector was integrated with a Waypoint Robotics Vector platform and performed an autonomous survey of storage containers with sealed radioactive sources. This work has been reviewed and issued as LANL LA-UR-19-31470.

Table of Contents

List of Tables xi

List of Figures xii

Chapter 1: Introduction 20

1.1 Los Alamos History and Background 20

1.2 Conceptual Framework 27

1.3 Summary 30

Chapter 2: Literature Review 32

2.1 Directional Neutron Detectors 32

2.2 Nuclear Robotics 43

2.3 Robot Control Methodology 55

2.4 Summary 58

Chapter 3: Methodology 60

3.1 Equipment and Software 62

3.1.1 Neutron Detector 63

3.1.2 MCNP 65

3.1.3 Mobile Platforms 67

3.1.3.1 Adept Pioneer LX 68

3.1.3.2 Waypoint Robotics Vector 70

3.1.4 ROS 72

3.1.5 Elastic Band Planning 72

3.1.6 Vector Peripherals 77

3.1.7 Testing Facilities 82

3.2 Strategy and Theory 85
3.2.1 Operational Constraints 85
3.2.2 Detector Aperture Equations 88
3.2.3 Neutron Sources 93
3.2.4 Neutron Shielding 97
3.2.5 Shielding Material 98
3.2.6 Counting Statistics 101
Chapter 4: Findings 105
4.1 Pioneer LX Approvals 105
4.2 Vector Approvals 108
4.3 Detector Characterization 111
4.3.1 Linear Dependence 112
4.3.2 Long Axis Rotation Dependence 114
4.3.3 Short Axis Rotation Dependence 117
4.3.4 Short Axis Rotation Dependence with Shielding 123
4.4 MCNP Configurations 128
4.5 Shield Characterization 139
4.6 Integrated Demonstration 146
4.7 Discussion 168
Chapter 5: Conclusions 170
5.1 Summary 170
5.2 Future Work 172
5.3 Concluding Remarks 174

Appendices....176
Appendix A....176
Appendix B....179
Appendix C....189
Appendix D....190
References....225

List of Tables

Table 1. Summary of mobile platforms used in radiological inspections (adapted from Tsitsimpelis et al., 2019). 54

Table 2. Transitional levels of autonomy (Brabec et al., 2011). 57

Table 3. Alpha decay and spontaneous fission rates for selected isotopes (Davis, Kornreich, & Lambert, 2011). 94

Table 4. Yields from (α, n) reactions (modified from Reilly et al., 1991) 95

Table 5. Average (α, n) neutron energy per target by source isotope (modified from Davis, Kornreich, & Lambert, 2011). 95

Table 6. Results from shield rotation test with source at 6 inches. 143

Table 7. Summary of Experiments Performed 171

List of Figures

Figure 1. Plutonium pit casting at LANL (Los Alamos National Laboratory, 2005).....21

Figure 2. PF-4 and surrounding buildings at Los Alamos National Laboratory circa 1995 (LANL, n.d.-b).....22

Figure 3. Hagan container showing “white powder” abnormality. 26

Figure 4. SAVY-4000 container lid showing corrosion. 26

Figure 5. Plan-view representation of multiple potential surveys within varying positions of a vault room, with radiological material storage on the walls. 29

Figure 6. Rough representation of an apertured neutron shield (green) around a detector (blue) with source nearby (yellow). 30

Figure 7. EVIDOS detector where A) shows the six detectors spread among a 30 cm polyethylene sphere and B) detail of the four layers of each detector (Luszik-Bhadra, d'Errico, Hecker, & Matzke, 2002).....34

Figure 8. Superheated emulsion detector with optical detection of bubbles (d’Errico, Fulvio, Maryanski, Selici, & Torrigiani, 2008). 35

Figure 9. Boron scintillator simulations for A) 10 keV, B) 100 keV, C) 1 MeV, and D) 10 MeV neutrons (Taylor, 2010). 36

Figure 10. A) Plastic scintillator fiber assembly (Imaida et al., 1999) and B) image of proton recoil for 65 MeV neutron (Ryan et al., 1999). 37

Figure 11. Large-area double-scatter fast neutron directional detector (Vanier et al., 2007). 38

Figure 12. Back projection of double-scatter neutron events to locate position of source (Vanier et al., 2007).....39

Figure 13. Smaller silicon wafer neutron collimators with larger Soller collimator in the middle (Cussen et al., 2001).....40
Figure 14. D20 neutron diffractometer layout (Hansen et al., 2008).....41
Figure 15. Measured and calculated neutron diffraction patterns for A) CRG and B) TRG; comparison of neutron and x-ray diffraction patterns of C) CRG and D) TRG; reconstructed apparent size of the particles from neutron patterns displayed along the <100> direction for E) CRG and F) TRG, and along the <001> direction for G) CRG and H) TRG (Sofer et al., 2014). 42
Figure 16. View of hot cells with operator using mechanical linkages (LANL, 2015).....43
Figure 17. ROVER system deployed to Three Mile Island (Gelhaus & Roman, 1990)....44
Figure 18. LOUIE I system deployed to Three Mile Island (Reilly et al., 1985). 45
Figure 19. Pioneer system deployed to Chernobyl (Tsitsimpelis et al., 2019). 46
Figure 20. RESQ robots for nuclear accident information collection (Kobayashi, Miyajima, & Yanagihara, 2002). 47
Figure 21. iRobot PackBot operating at Fukushima (TEPCO, 2019).....48
Figure 22. Close-up view of radiation meter on PackBot at Fukushima (TEPCO, 2019). 48
Figure 23. JAEA-3 system for gamma imaging that was created from an earlier RESQ robot (Kawatsuma et al., 2012).....49
Figure 24. PMORPH robot to inspect the Unit 1 primary containment vessel at Fukushima (TEPCO, 2019). 50
Figure 25. Khepera II mobile robot for radiation mapping (Cortez et al., 2009). 51
Figure 26. Mobile robot to map neutron and gamma dose rates for cyclotron beam line (Ravishankar et al., 2013). 52

Figure 27. RadBot gamma mapping system on a Pioneer 3-DX base (von Frankenberg et al., 2012). 53
Figure 28. Illustrative representation of the final system able to drive around a room and scan locations for radiation. 62
Figure 29. Internal view of neutron detector. 63
Figure 30. Total and (n, α) cross sections for boron-10 (KAERI, n.d.). 64
Figure 31. Tabular representation of MCNP6 particle types, energy ranges, and interaction physics (T. Goorley et al., 2012). 66
Figure 32. The Adept Pioneer LX mobile robotic platform (Adept MobileRobotics, 2013). 69
Figure 33. Pioneer MobileEyes software showing the scanned map of uncleared location at LANL (created with Mapper3) and the range of the laser scanner in blue. 70
Figure 34. Waypoint Robotics Vector mobile platform at LANL. 71
Figure 35. A) A path generated by a planner; B) applying both an internal contraction force and an external repulsion force; C) a new moving obstacle is introduced; D) the path is deformed to avoid the obstacle (Quinlan & Khatib, 1993). 73
Figure 36. Inflation of costmap values as related to robot footprint areas (Marder-Eppstein, Lu, & Hershberger, 2018). 74
Figure 37. 2-D costmap visualization for Vector robot. 76
Figure 38. Visual simulation of the environment that correlates to the costmap in the previous figure. 76
Figure 39. Universal Robots UR5e arm (Universal Robots, 2018). 78
Figure 40. Optris thermal camera (Optris, 2015). 79

Figure 41. FLIR visual camera (FLIR, 2019). 79
Figure 42. Visual and thermal cameras bolted to arm end-effector. 80
Figure 43. Vector platform integrated with arm and sensors. 81
Figure 44. Pioneer LX mobile platform located in PF-4. 83
Figure 45. View of testing hallway at TA-35. 84
Figure 46. Alternate view of testing hallway at TA-35. 85
Figure 47. Solid angle subtended by detector from the source. 88
Figure 48. Parallel plane source and detector geometry (Tsoulfanidis & Landsberger, 2011). 90
Figure 49. Point source with circular aperture detector geometry (Tsoulfanidis & Landsberger, 2011). 91
Figure 50. Neutron energy spectra on plutonium and curium isotopes (Pérot et al., 2018). 96
Figure 51. Neutron energy spectrum of Cf-252 spontaneous fission (Park, 2003). 97
Figure 52. Monte Carlo simulations of 1 MeV neutrons entering cylinders of polyethylene, aluminum, and lead (Rinard, P., n.d.). 99
Figure 53. Total cross sections for two most abundant isotopes of boron, lithium, and cadmium (KAERI, n.d.). 100
Figure 54. Distributions expected for the net counts N_S for the cases of A) no activity present and B) real activity is present. L_C is the critical level or "trigger point" of the counting system, and L_D corresponds to the mean number of net counts needed for MDA (modified from Knoll, 2000). 103
Figure 55. Container shelving in Pioneer LX room. 107
Figure 56. Machined aluminum base and arm mounting plate. 109
Figure 57. Detector and shield assembly mounted on front of base plate. 110

Figure 58. Partially constructed shield mounted on base plate showing the side detail of shield mounting bracket.....111
Figure 59. Linear dependence experimental setup. 113
Figure 60. Count rate results from linear dependence test.....114
Figure 61. Detector rotation and source positions for long axis rotation dependence test. 115
Figure 62. Detector and source in position 1 for long axis rotation dependence test. 115
Figure 63. Count rate results for long axis rotation dependence test in position 1. 116
Figure 64. Count rate results for long axis rotation dependence test in position 2. 117
Figure 65. Detector rotation and source positions for short axis rotation dependence test. 118
Figure 66. Detector at 0 degrees and source at position 5, 140 cm, for short axis rotation dependence test. 118
Figure 67. Detector at the 40 degrees to the right position and source at position 5, 140 cm, for short axis rotation dependence test. 119
Figure 68. Count rate results for short axis rotation dependence test at position 3 (75 cm). 120
Figure 69. Count rate results for short axis rotation dependence test at position 4 (100 cm). 121
Figure 70. Count rate results for short axis rotation dependence test at position 5 (140 cm). 122
Figure 71. Normalized count rate results for short axis rotation dependence tests at positions 3, 4, and 5. 123
Figure 72. Count rate results for short axis rotation dependence test at position 4 with 5 cm of 2% borated polyethylene. 124

Figure 73. Count rate results for short axis rotation dependence test at position 4 with 10 cm of 2% borated polyethylene. 125
Figure 74. Count rate results for short axis rotation dependence test at position 4 with 5 cm of polyethylene.....126
Figure 75. Count rate results for short axis rotation dependence test at position 4 with 10 cm of polyethylene.....127
Figure 76. Normalized count rate results for short axis rotation dependence at position 4 with varied shielding configurations.....128
Figure 77. MCNP model of shield, inner void, and two sources (Cf-252 and diffuse incoming background) that can be adjusted.....130
Figure 78. Base example for MCNP model.....131
Figure 79. MCNP model with polyethylene thickness adjusted.....131
Figure 80. MCNP model with cadmium layer adjusted. 132
Figure 81. MCNP model with wedge window angle adjusted. 132
Figure 82. MCNP model with three-layer shield and bounding region.....134
Figure 83. Close-up view of MCNP model with three-layer shield. 135
Figure 84. CAD model of three-layer shield. 136
Figure 85. CAD drawing of outer borated polyethylene shield layer.....137
Figure 86. CAD drawing of middle cadmium sheeting layer.....138
Figure 87. CAD drawing of inner HDPE layer.....139
Figure 88. Fabricated neutron shield and detector in test location. 140
Figure 89. Alternate view of test location with shield, detector, and source stand. 141
Figure 90. Detector count rate with source at 6 inches and associated count rate errors. 144
Figure 91. Variations in detection efficiency with varied source heights.....145

Figure 92. Storage cabinet 1 of integrated system test. 148
Figure 93. Storage cabinet 2 of integrated system test. 149
Figure 94. Storage cart 1 of integrated system test. 150
Figure 95. Drum 1 of integrated storage test. 151
Figure 96. Drum 2 of integrated system test. 152
Figure 97. The integrated system test scanning locations of A) Storage cabinet 1 (red), B) Drum 1 (orange), C) Cart 1 (green), D) Storage cabinet 2 (blue), and E) Drum 2 (pink). 153
Figure 98. Notional path traveled (not to scale) for integrated system test with locations A) Storage cabinet 1 (red), B) Drum 1 (orange, C) Cart 1 (green), D) Storage cabinet 2 (blue), and E) Drum 2 (pink). 154
Figure 99. Neutron count rate during integrated system test with room positions indicated. 155
Figure 100. Scanning room at locations A) Storage cabinet 1, B) Drum 1, C) Cart 1, D) Storage cabinet 2, and E) Drum 2. 156
Figure 101. Image collected from visual camera pointing towards drum 2 while platform is located at drum 1. 157
Figure 102. Visual camera image of container with Cf-252 source in storage cabinet 2. 158
Figure 103. Visual camera image of container with broken sealing tape in storage cabinet 1. 159
Figure 104. Visual camera image of left side of bottom row of storage cabinet 2. 159
Figure 105. Visual camera image of container with simulated "white powder" on lid in storage cabinet 1 with flashlight attached. 160

Figure 106. Visual camera image of container with broken sealing tape and SAVY-4000 container below it in storage cabinet 1 with flashlight attached. 160
Figure 107. Visual camera image of containers in storage cabinet 2 with flashlight during execution of base platform movement to storage cabinet 2. 161
Figure 108. Visual camera image of dented container in storage cabinet 2 with flashlight. 161
Figure 109. Thermal camera image (left) with related photo (right) of view down testing hallway at TA-35.....162
Figure 110. Thermal image of empty cans in storage cabinet 2. 162
Figure 111. Thermal camera image of A) Pu-239 source (left) next to empty container (right) in storage cabinet 1, B) Cf-252 source in storage cabinet 2, and C) Pu-238 heat source next to drum 2. 163
Figure 112. A) Thermal camera image of Pu-238 source in 5 gallon container on cart 1 and B) Visual camera image of the same source and container. 164
Figure 113. Histogram of temperatures measured from thermal camera image of Pu-238 source in 5 gallon container on cart 1. 165
Figure 114. Thermal camera image of dented and Pu-238 containers in storage cabinet 2 during execution of base platform movement to storage cabinet 2.....166
Figure 115. Thermal camera images showing thermal reflection of Pu-238 container in Storage cabinet 2 left door as arm pans to the right from A) Storage cabinet 2 left door (with the reflection), to B) Storage cabinet 2 left door and dented container, to C) dented container and left side of Pu-238 container, and to D) the actual Pu-238 container.....167

Chapter 1: Introduction

The nuclear weapons complex in the United States has dutifully served the nation in its mission of stockpile stewardship and maintenance of the nuclear deterrent. However, the capabilities of the aging facilities used by the complex are often at odds with the need to adequately maintain the stockpile. In order to succeed, unique technological solutions must be employed to ensure the safety and security of the complex while meeting production goals. This work proposes one such solution regarding the safe and efficient operation of radiological facilities storing nuclear materials at Los Alamos National Laboratory (LANL).

1.1 Los Alamos History and Background

Los Alamos National Laboratory (LANL) was founded in 1943 amidst the secrecy of the Manhattan Project. World War II adversaries posed many dangers to the Allies, and work on nuclear weapons commenced to help counter the threat of the Nazis developing their own atomic bomb. Los Alamos, New Mexico, was selected as the site (originally termed "Site Y") to host the scientists and engineers, led by J. Robert Oppenheimer, who would design and build the Fat Man and Little Boy nuclear devices (LANL, n.d.). In the years since the Manhattan Project ended, LANL has continued to play a prominent role regarding the design and maintenance of nuclear weapons. A nuclear weapon works under the basic principles of nuclear fission and nuclear fusion. In these reactions, an atomic nucleus is either split or joined with another nucleus, and in the process it releases a tremendous amount of energy. The main part of a nuclear weapon that undergoes fission is the "pit," and it is this core part that is responsible for generating the explosion. The pit

is made from plutonium-239 due to the isotope's ideal properties for the nuclear reactions powering the weapon (Figure 1 displays an operation to cast plutonium).

Figure 1. Plutonium pit casting at LANL (Los Alamos National Laboratory, 2005).

Plutonium pits are manufactured at LANL in Plutonium Facility 4 (PF-4) within Technical Area 55 (TA-55) (Figure 2). PF-4 was built in the early 1970s and became operational in 1978 (LANL, n.d.). It was originally built as a research facility. Today it is the nation's last remaining high-security, multi-purpose plutonium facility, and is used to

perform both actinide-related research and production. The two-level structure houses many gloveboxes on its first floor plus a vault and other storage areas for nuclear material in the basement.

Figure 2. PF-4 and surrounding buildings at Los Alamos National Laboratory circa 1995 (LANL, n.d.-b).

The basement vault and other storage locations are key to facility operations. While some stored items are mostly historical in nature, many reflect the key role the vault plays in support of current production. As plutonium is processed into a pit, it generates other materials that must either be processed immediately or placed into the vault awaiting later processing or disposition. Having recently surpassed its fortieth anniversary of operation,

it should be no surprise that the PF-4 vault and related storage areas could be relatively full. Proper management of the vault is vital to the successful production of pits. Elevating this need even higher is the congressional mandate that LANL increase pit production to 30 pits per year by 2026 with contingency planning to potentially produce 80 pits per year beyond 2030 (National Defense Authorization Act, 2018).

Because of its nature as a storage location for some highly radioactive items, operations in the vault pose unique safety considerations. The vault is organized into several concrete rooms off a main corridor. The rooms generally contain shelving that houses several rows of radioactive items. The dose rate within rooms can be high. It is possible of for a worker who performs many vault operations to reach their annual maximum allowable dose of 2,000 mrem before a full year has passed. (2,000 mrem per year is the local LANL limit and well below under the Department of Energy [DOE] limit of 5,000 mrem, as described in 10 C.F.R. § 20 as a safety margin.) Complicating matters is the presence of some legacy containers of nuclear materials that are being remediated as part of the Material Recycle and Recovery (MR&R) Program. These hazards necessitate the use of respirators in some locations to prevent the inhalation of radioactive contamination in the event that a legacy container is accidentally breached. However, respirators are bulky, reduce visibility, and ultimately can increase the time required to perform a task in the vault. It is thus desirable to reduce time spent in the vault when possible to reduce the dose to the worker. The federal policy of reducing dose where possible is called ALARA, or "As Low As Reasonably Achievable," and is formalized in 10 C.F.R. § 20. The federal code decrees that principle means "making every reasonable effort to maintain exposures to radiation as far below the dose limits . . . as is practical consistent with the purpose for which the licensed activity is undertaken, taking into account the state of technology, the economics of improvements in relation to state of

technology, the economics of improvements in relation to benefits to the public health and safety, and other societal and socioeconomic considerations, and in relation to utilization of nuclear energy and licensed materials in the public interest." The basic methods for implementing ALARA are to reduce the time spent present in radiation, increase the distance from radiation, and add shielding to reduce the amount of radiation reaching a person.

In order to keep the dose for vault workers at manageable levels, time spent in the vault is kept to a minimum, but this prevents the collection of frequent survey information about the stored items. Dose survey maps are generated about once per quarter, yet items are moved around with greater frequency, which means that the maps are of declining use as the radiation landscape changes. Inventory checks are similarly limited. This work suggests a way to improve the acquisition of data to better monitor the health of the vault's collection while reducing workers' dose by automating various stored material inspection tasks.

There are other practical benefits to increased surveillance of items stored in the radiological vault. Reduction of worker dose is a strong and worthy driver of vault improvements; indeed, early motivations for the redesign of LANL nuclear material storage containers to a more robust form included a concern for worker safety (as opposed to other motivations such as material security). For example, although the more modern SAVY-4000 storage containers have replaced older containers as the gold standard, the Hagan container was created by a vault supervisor to better protect his vault operations staff from radiological exposure (K. Veirs, personal communication, April 26, 2018). Increased knowledge about the state of stored materials and their containers could be used to detect minor issues before they transform into major problems.

Nuclear safeguards are measures implemented to protect against diversion and proliferation of nuclear material. In part to support nuclear safeguards, information is stored in a database to accurately track and account for stored nuclear materials. This database records information such as the isotopes, masses, and chemical forms of the nuclear materials. In support of safeguards, nondestructive assay (NDA) techniques or analytical chemistry is performed to verify the masses. However, these tests are performed only as needed, and given that LANL's facility has been operational since the 1970s, it is possible that numerous items have not been inspected or measured for years or decades. The database tracking system can perform decay calculations to accommodate the passage of time on the nuclear masses, but little information is able to be recorded about the possible dynamic changes happening over time. Abnormal conditions are not unheard of for stored nuclear materials. Although container surveillance is performed annually on a mix of 15 Hagan and SAVY-4000 containers, there have been several instances where a worker went into the PF-4 vault to retrieve a container and just happened to see another nearby container that had signs of damage.

Figure 3. Hagan container showing “white powder” abnormality.

Figure 4. SAVY-4000 container lid showing corrosion.

It is not currently feasible to merely increase the number of container surveillances performed per year because of the associated increase in personnel time, personnel dose, and other associated facility resources, including gloveboxes and radiological control technicians (RCTs). However, if a robotic system were to frequently survey the vault or other storage locations, then a day-to-day baseline could be established and used to more rapidly detect possible abnormal conditions. Examination of more frequently collected information over time may also have the potential to reveal changes that are not initially perceptible to human operators.

1.2 Conceptual Framework

Materials stored in the vault vary, but they include large amounts of Special Nuclear Material (SNM). Special Nuclear Material is defined by Title I of the Atomic Energy Act of 1954 as plutonium and uranium enriched with uranium-233 or uranium-235. Due to the sensitive nature of such materials, the descriptions of their types and quantities will remain generalized in this work. However, this is not to say that the inventory is not well characterized. Materials that are held under the accountability of safeguards are tracked in an internal database. It is a fair assumption that the vault holds some amount of plutonium oxides and metal, among other things. An idealized version of the vault will be substituted in this work, where needed, using arbitrary percentages for material composition so as to not divulge sensitive information. A fully deployed automatic surveying system could utilize more specific information about the items within the vault.

Unexpectedly high dose readings sometimes have been measured in certain sections of a room. In the hypothetical case of such an incident, it is possible that the

abnormal readings could have been detected by a robotic survey system much earlier than they would have been by human workers. However, a system that merely creates a map without a way to identify the source is self-limiting. If one—if not the most important—of the major motivations behind implementing a robotic system for creating dose maps is the reduction in worker dose, then what good is a system that requires a human to further investigate unexpectedly high doses? It would be far better for a system to identify where a lone source is located within a room; it could locate the source by following the strength of the detection signal over distance. Yet the vault does not have the luxury of easily distinguishing between sources due to the sheer amount of individual points of radiation generation and the subsequently scattered general radiation environment. The vault is a veritable sea of radiation. Needing to involve vault workers to fish out which item is causing higher doses would just expose them to the very radiation a robotic mapping system was intended to reduce. Thus, it is imperative that any robotic vault survey system also be equipped to source the origination of detected radiation. This work seeks a method to develop such a capability.

Current dose maps are planar representations of collective dose rates from radiation incident at a point from all directions in three dimensions. While dose and flux are not the same thing, there is a definite correlation, and published American National Standards Institute (ANSI) fluence-to-dose factors can be used to calculate dose from flux. Thus, this work will focus on comparisons of neutron count rates as a measure of the flux. A robotic system could easily replicate a human worker walking around a vault room holding a detector to make a dose map. In order to be able to distinguish between sources contributing to the dose at one point, there would need to be a way to determine some sort of directionality of the incoming radiation. In other words, if one were able to separate out the angular dependence of the flux as a function of position within a room, it would be

possible to gain some measure of the direction from which the detected radiation was coming. Piecing together many such measurements in a room would alleviate the problem of trying to distinguish stored radioactive items from one another. To achieve that goal, a detector can be mounted upon a robot that positions the detector in various locations of the room while also being able to orient the detector at those locations (Figure 5).

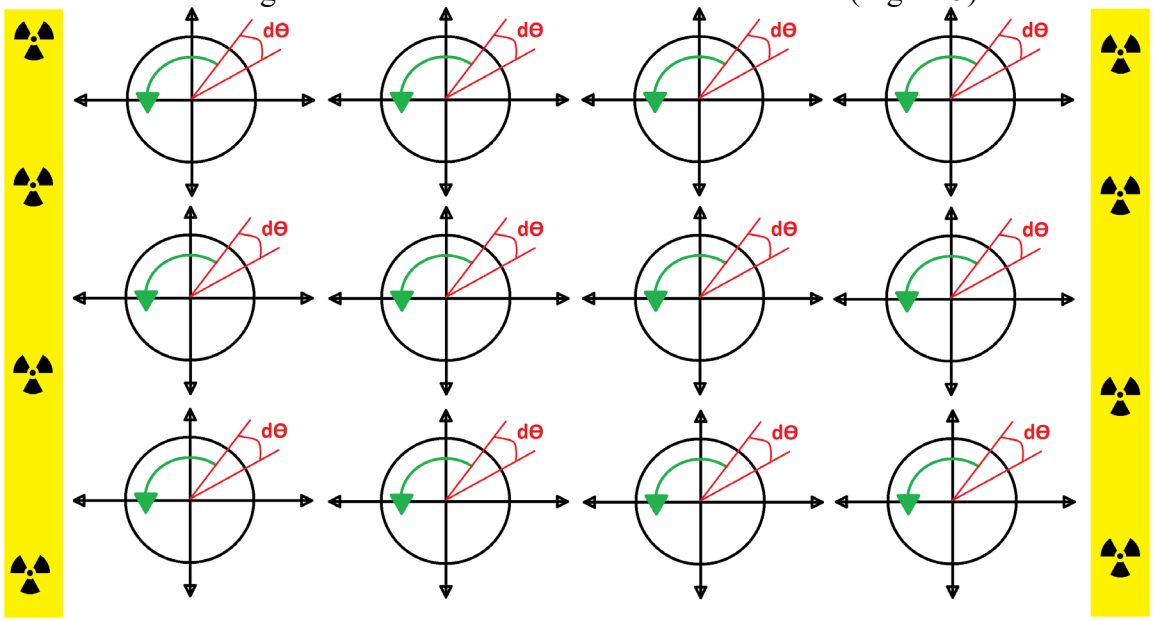


Figure 5. Plan-view representation of multiple potential surveys within varying positions of a vault room, with radiological material storage on the walls.

This process can be thought of as analogous to using a maritime spyglass. A spyglass is a handheld, monocular device often associated with sailors and pirates of earlier times looking for dry land while at sea. While an important function of a spyglass was visual magnification, it is the long barrel, which precludes light other than the light directly down the line of sight from reaching the operator, that is analogous to the radiation sourcing

problem. A detector that can effectively look through a radiation spyglass at individual storage locations while shielding radiation from other sources will be able to get a measure of how much of the total radiation at a position in a vault room is due to that storage location. The plan is to investigate the efficacy of placing a neutron shield around most of the detector that also contains an aperture to allow some radiation in. A simplified model is shown in Figure 6, where the green component is the shield around the blue detector with radiation emitted by the yellow source.

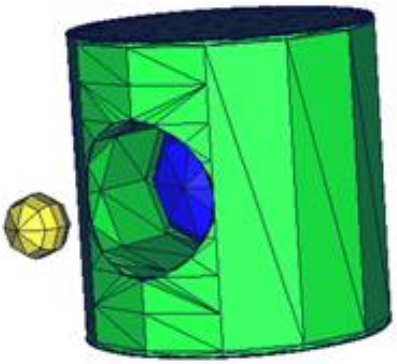


Figure 6. Rough representation of an apertured neutron shield (green) around a detector (blue) with source nearby (yellow).

1.3 SUMMARY

This work will explore the theoretical framework and justification for the process of creating an artificial aperture around a neutron detector and its incorporation into a mobile robotic platform for radiation mapping. Then, a literature review will explore similar work that has been done both for biasing of neutron detection directionality as well

as the incorporation of detectors with robotics and associated radiation mapping. The experimental methods chapter details what equipment and methodology are used, while the results chapter describes the outcomes from modeling and implementation. Finally, the conclusions chapter summarizes this work and posits ideas for future work.

Chapter 2: Literature Review

This chapter summarizes some of the previous work that has been done regarding directional neutron detectors and radiation surveying with robotics. While most applications of detectors on robots focus on others kinds of radiation, there have been cases where a neutron detector has been incorporated with a system. Part of the novelty of this work is the incorporation of a *directional* neutron detector with a robotic system.

2.1 Directional Neutron Detectors

A useful starting point would be the review paper of Balmer, Gamage, and Taylor (2014) entitled "Critical review of directional neutron survey meters." Balmer et al. argue that the standard practice of measuring the ambient dose equivalent when surveying areas tends to lead to misleading dose estimates and over-restrictive work practices. The greatest dose risk is from radiation incoming from an antero-posterior direction, but if the radiation is coming from the postero-anterior direction, the effective dose will be smaller. Similarly, personal dosimeters are worn on the front of the torso, and using these tools to estimate dose risk can underestimate the risk from a postero-anterior direction due to the shielding effects of the body itself. The authors argue that improved directional neutron survey meters would lead to better estimates of dose and understanding of work hazards. Existing neutron survey techniques generally rely on single detectors. Spectroscopy can be performed for thermal neutrons using Bonner spheres of varying sizes around the detector. However, this technique is time-consuming, and any directional detector that would be used in daily surveys needs to be easily usable and portable.

The "Evaluation of Individual Dosimetry in Mixed Neutron and Photon Radiation Fields" or EVIDOS project created a multi-detector system to measure directionality. Six detectors were placed around a 30 cm polyethylene sphere. Each detector was constructed

from four layers of silicon detectors that were optimized for different energy levels. The layers for lower-energy neutrons were ^{6}LiF coated silicon diodes surrounded by boron-doped plastic. The layers for higher-energy neutrons were surrounded by a polyethylene layer to detect charged particles for proton recoil events. This system (Figure 7) then examined the response of all 24 detecting elements to examine the neutron spectrum. The drawbacks of this system are that the neutron spectrum had to be known beforehand to understand the data coming from the detecting elements, and the 30 cm polyethylene sphere is somewhat heavy at 13.6 kg.

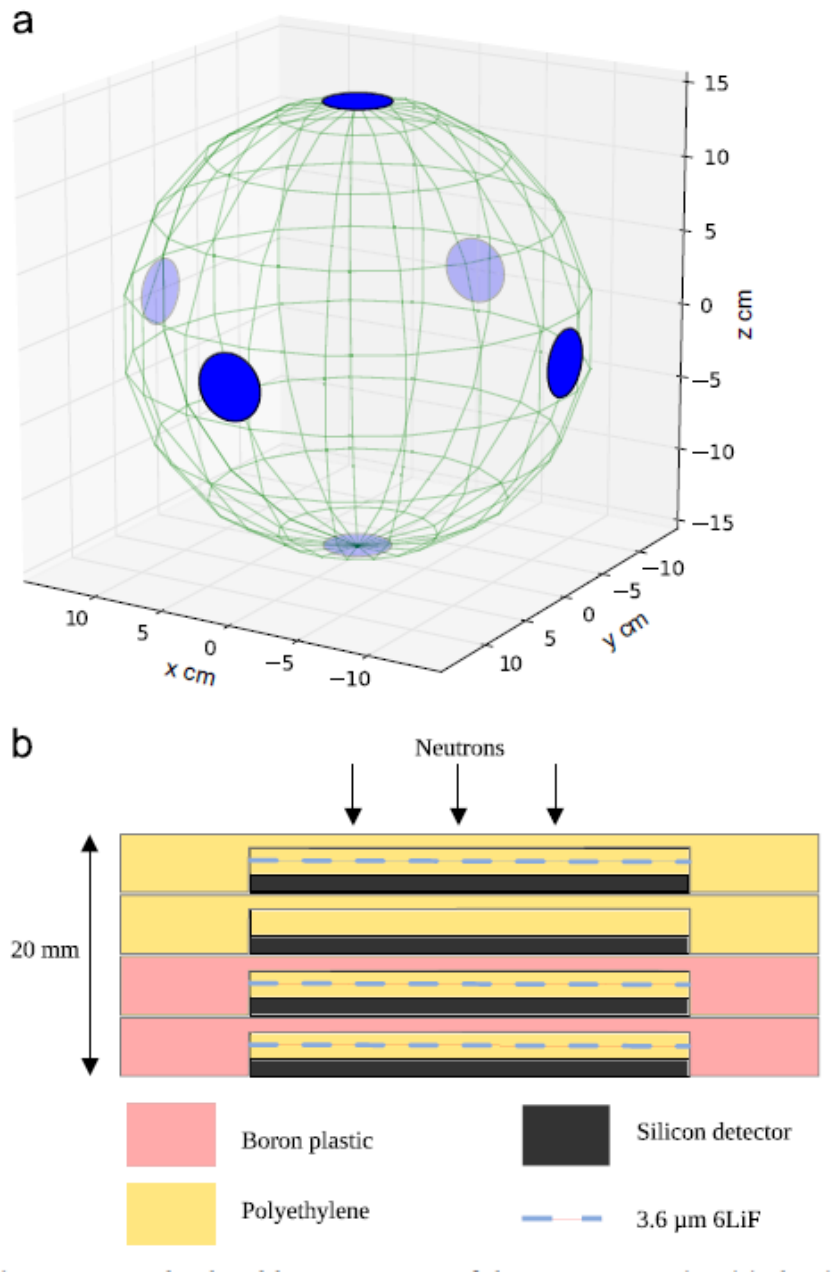


Figure 7. EVIDOS detector where A) shows the six detectors spread among a 30 cm polyethylene sphere and B) detail of the four layers of each detector (Luszik-Bhadra, d'Errico, Hecker, & Matzke, 2002).

Some work has also explored the use of superheated emulsions for directionality detection. In one setup, a 30 cm sphere (nylon) surrounds a superheated emulsion of dichlorotetrafluoroethane. When neutrons enter the inner part with the superheated emulsion, bubbles will form. These bubbles can be measured by acoustic piezodetectors (d'Errico, Giusti, Reginatto, & Wiegel, 2004) or LEDs (using a cylindrical detector geometry) that emit light that scatters off the bubbles into various photodiodes around the emulsion (Figure 8). This approach does not easily enable portability because of the requirement to tightly control the temperature of the emulsion.

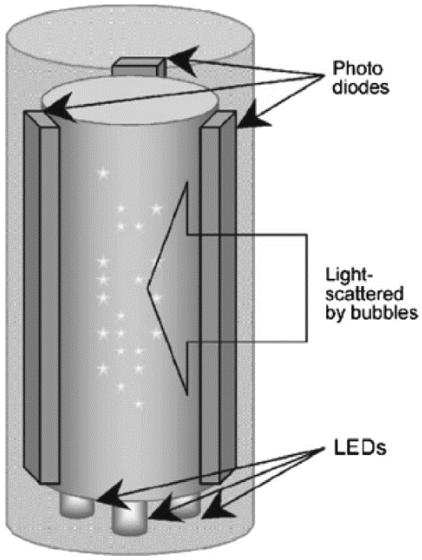


Figure 8. Superheated emulsion detector with optical detection of bubbles (d'Errico, Fulvio, Maryanski, Selici, & Torrigiani, 2008).

Taylor (2010) investigated the construction of a directional detector with Monte Carlo N-Particle Transport Code (MCNP) using a 20.32 cm sphere of a borated scintillator that is interrogated by a tetrahedral arrangement of photomultiplier tubes (Figure 9). Neutrons entering the scintillator cause proton recoil events. Results from

simulation were analyzed with a neural network to learn the distributions for given neutron energies.

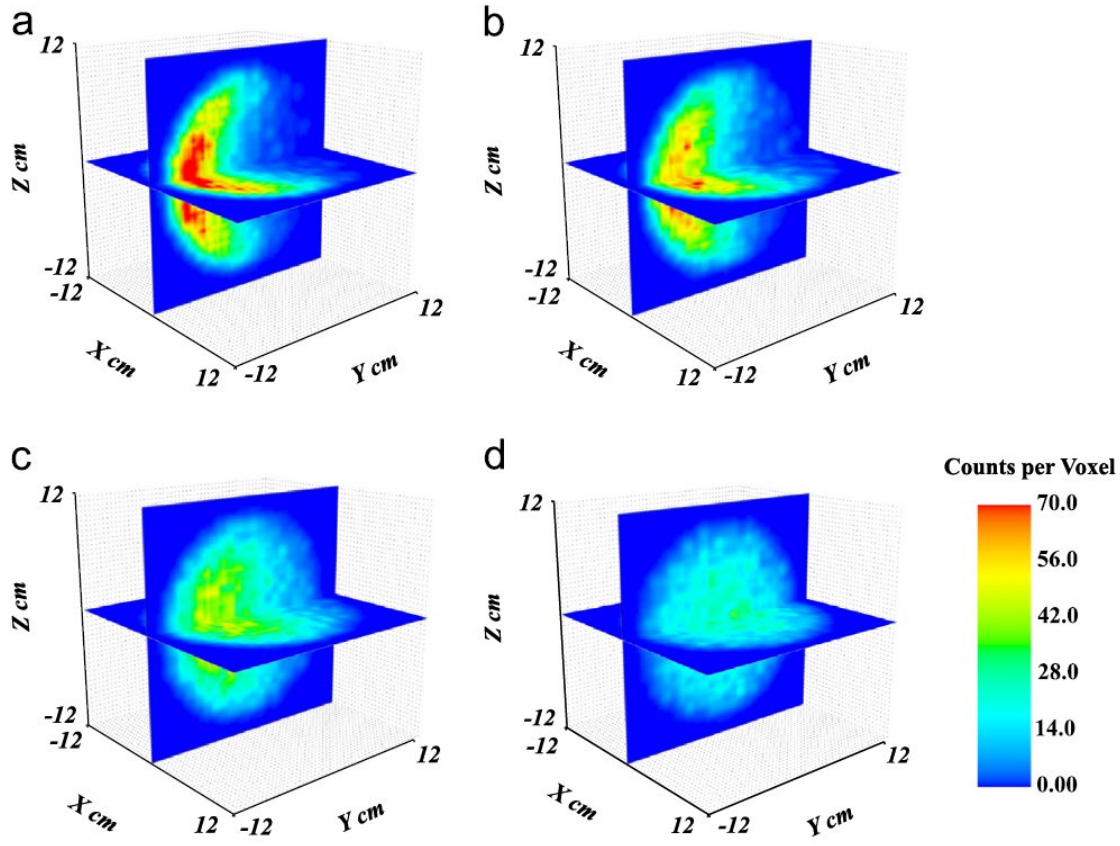


Figure 9. Boron scintillator simulations for A) 10 keV, B) 100 keV, C) 1 MeV, and D) 10 MeV neutrons (Taylor, 2010).

As alluded to in some of these detector designs, proton recoil instruments can measure neutron energy from the measured energy and direction of recoiled protons after collision in a thin layer of hydrogenous material (Peurrung, 2000). However, this technique is more typically useful for high-energy neutrons (resolution of 1%–3% for 14.1 MeV neutrons). Scintillating plastic fibers can also be used for this technique, but, again, usefulness is generally limited to high-energy neutrons (Figure 10).

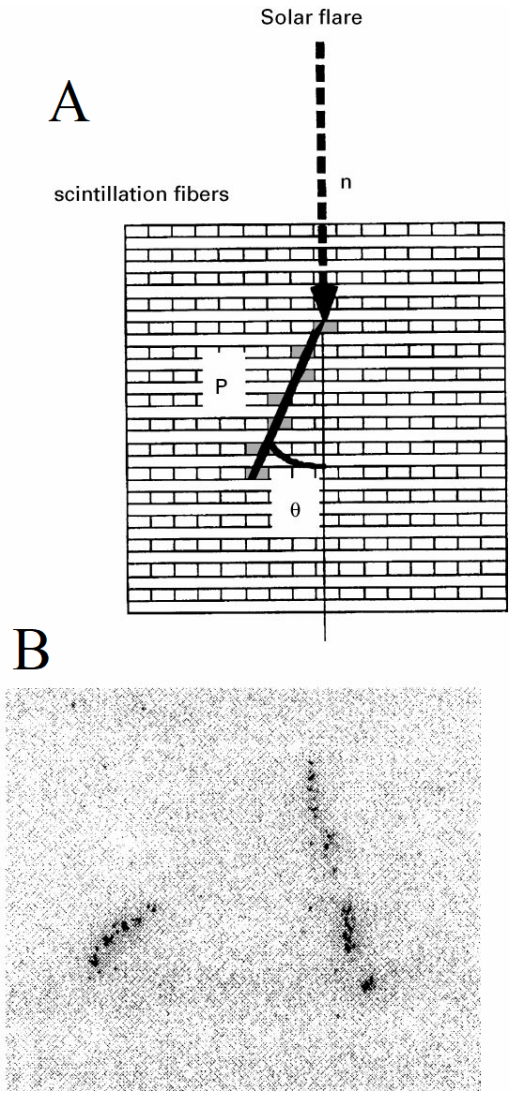


Figure 10. A) Plastic scintillator fiber assembly (Imaida et al., 1999) and B) image of proton recoil for 65 MeV neutron (Ryan et al., 1999).

Vanier, Forman, Dioszegi, Salwen, and Ghosh (2007) expanded upon the proton recoil detector by creating a double proton recoil detector with two planes of plastic scintillators (four front and four back paddles vertically stacked with photomultiplier tubes at the ends of each paddle). This system can locate the position of an event in the horizontal

direction and can detect fission energy neutrons, though because of its large size, it is not easily portable (Figure 11 and Figure 12).

Figure 11. Large-area double-scatter fast neutron directional detector (Vanier et al., 2007).

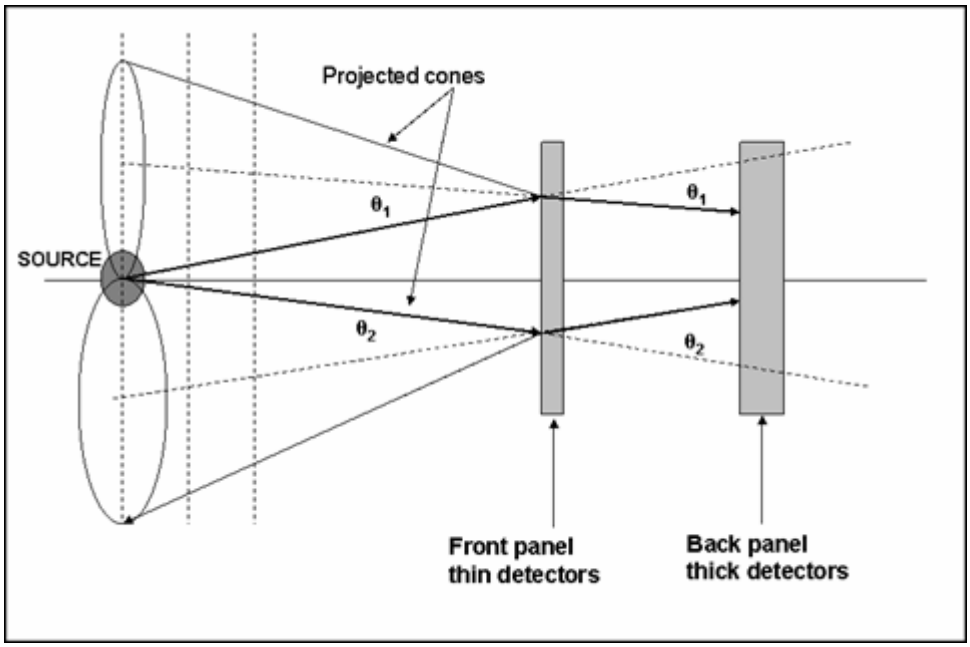


Figure 12. Back projection of double-scatter neutron events to locate position of source (Vanier et al., 2007).

Systems for determining directionality for neutron detection often want to *measure* the direction of incoming neutrons, but there are other relevant applications that seek to *control* the direction of neutrons. Early work for neutron spectrometry developed the use of Soller-type slit collimators for neutrons. Soller collimators are stacks of closely packed thin parallel foils. Materials can vary, but modern ones are typically made of 80 μm–125 μm thick gadolinium oxide, boron-10 coated polyethylene terephthalate (PET), or Kapton foil (JJ-XRAY, 2019). Caglioti and Casali (1962) suggested the use of thin rubber sheets similarly stacked as an inexpensive and easy-to-manufacture alternative. Friedmann and Rauch (1970) introduced the use of a curved Soller collimator with nickel-coated channels to focus neutrons from a reactor beam line. These style of collimators still see use today, but single crystal silicon wafers coated with multilayers have also been explored as collimators. The silicon wafers have a very small amount of absorption and incoherent

scattering and thus act as transmitting channels for neutrons. The multilayers are two reflecting layers around an inner absorbing layer (often gadolinium) (Cussen, Hoghoj, & Anderson, 2001). Silicon wafer collimators are smaller in size than Soller types (Figure 13).

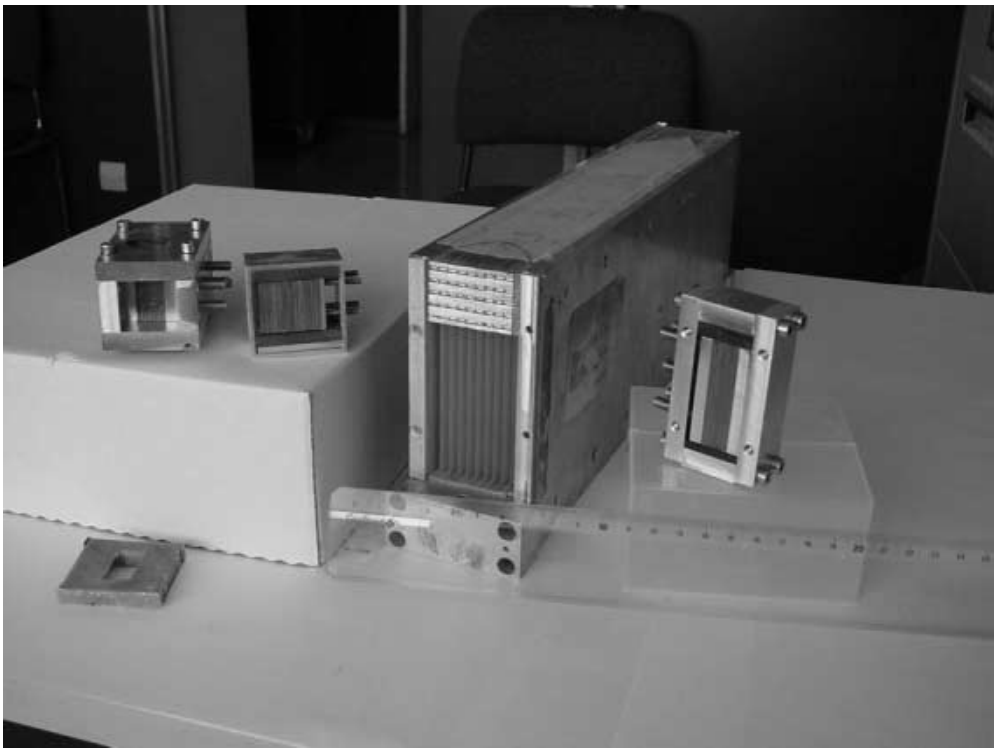


Figure 13. Smaller silicon wafer neutron collimators with larger Soller collimator in the middle (Cussen et al., 2001).

Collimators are an essential component of neutron diffractometers used to explore the atomic structure of materials. One example is the D20 instrument at the Insitut Laue-Langevin, which is a two-axis diffractometer capable of a neutron flux of 10^8 neutron/sec/cm^2 (Hansen, Henry, Fischer, Torregrossa, & Convert, 2008). The layout of this diffractometer is shown in Figure 14.

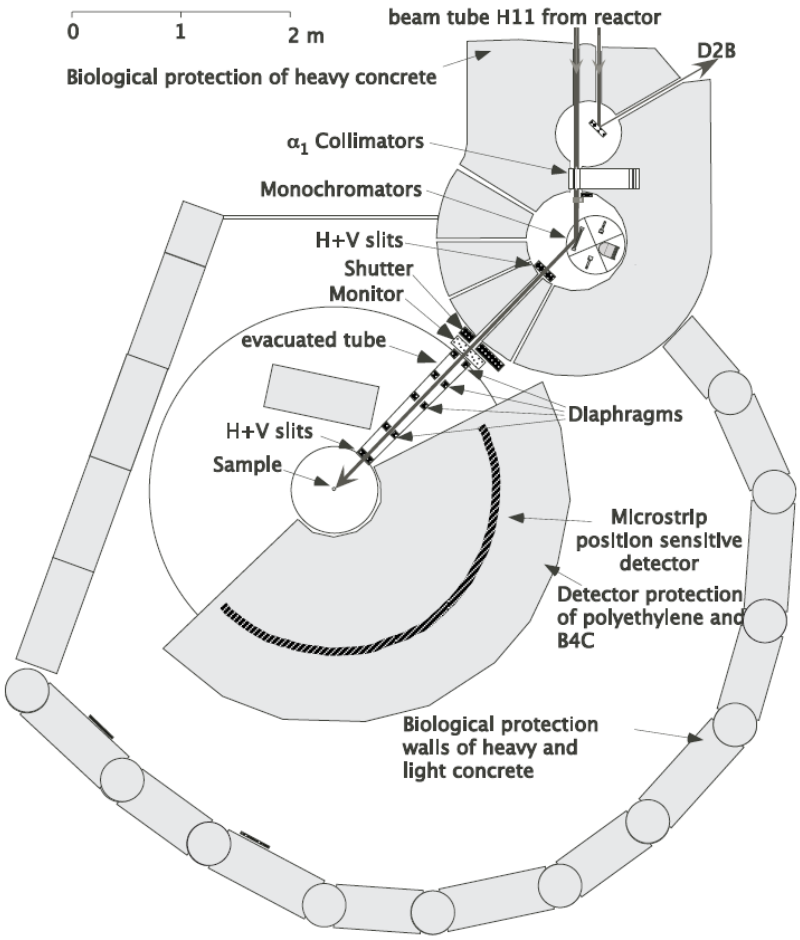


Figure 14. D20 neutron diffractometer layout (Hansen et al., 2008).

An example of results from neutron diffraction can be seen in Figure 15. These results are the measured and calculated neutron diffraction patterns for chemically reduced graphene and thermally reduced graphene as measured from the MEREDIT diffractometer (Sofer et al., 2014).

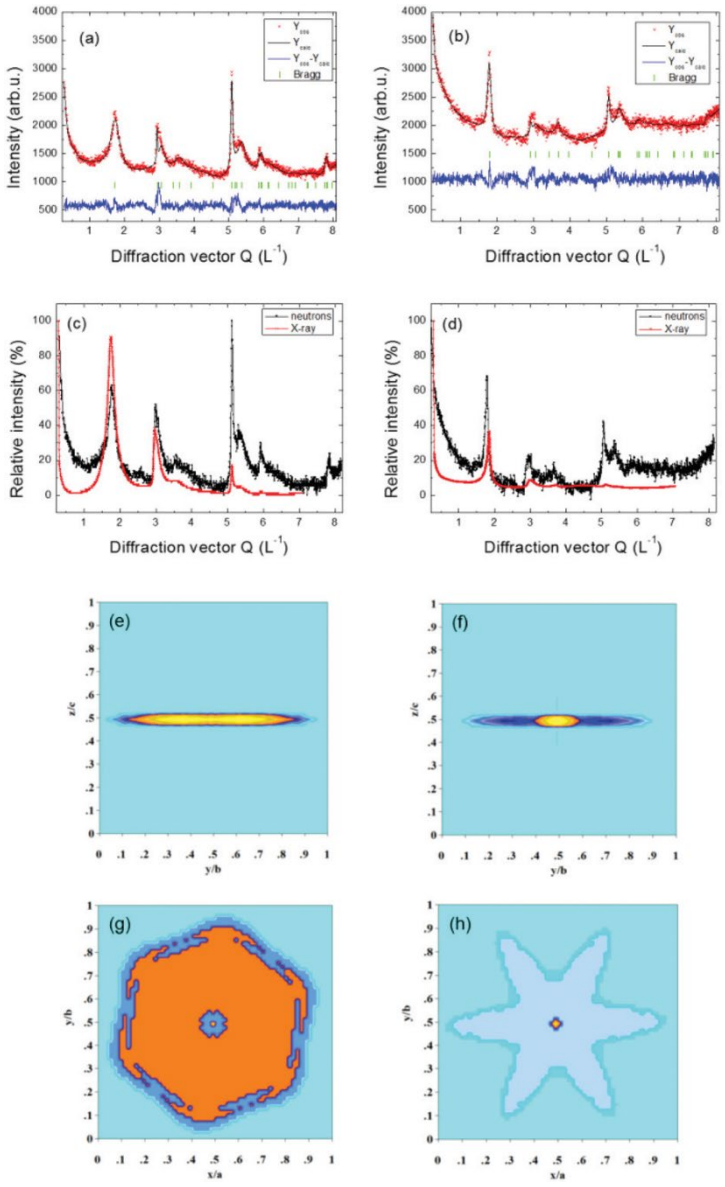


Figure 15. Measured and calculated neutron diffraction patterns for A) CRG and B) TRG; comparison of neutron and x-ray diffraction patterns of C) CRG and D) TRG; reconstructed apparent size of the particles from neutron patterns displayed along the <100> direction for E) CRG and F) TRG, and along the <001> direction for G) CRG and H) TRG (Sofer et al., 2014).

2.2 Nuclear Robotics

Robotic technologies have long been incorporated into the nuclear industry. Extreme radiation has necessitated the use of machines for remote handling. Hot cell manipulators are an early example of technology used to increase the distance between humans and radioactive sources to protect operators, and their series of mechanical linkages is quite evocative of robotic designs (Figure 16).

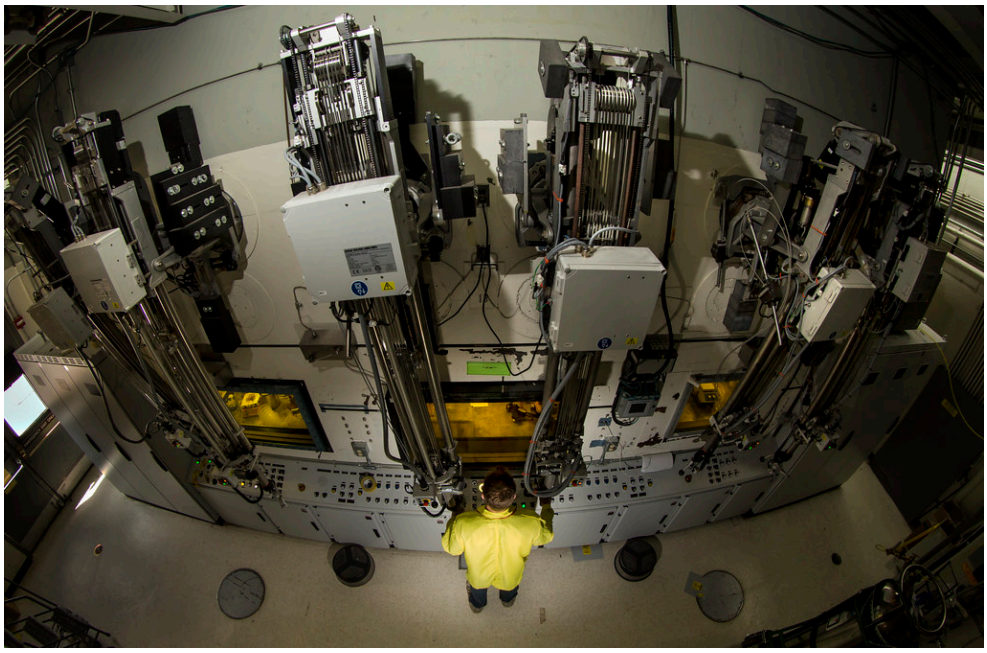


Figure 16. View of hot cells with operator using mechanical linkages (LANL, 2015).

An early example of a mobile robot being used to survey radiation was a platform used after the partial reactor meltdown at Three Mile Island (TMI-2) in 1979. However, the robot was not used until four years after the accident (Tsitsimpelis, Taylor, Lennox, & Joyce, 2019). Three different systems were used in the clean-up process for TMI-2. ROVER (Figure 17) was a six-wheeled platform that was remotely operated and used for

environmental monitoring, video transmission, sampling, and decontamination. Three ROVER systems were manufactured; two were deployed to the basement of Unit 2 at TMI and the third was used as a training and testing device. The two other systems used for clean-up at TMI were the LOUIE I and LOUIE II systems. LOUIE I (Figure 18) was primarily used for surveying radiation, as its smaller footprint enabled it to reach areas larger platforms couldn't. LOUIE II was used to cut pieces of concrete (Tsitsimpelis et al., 2019).

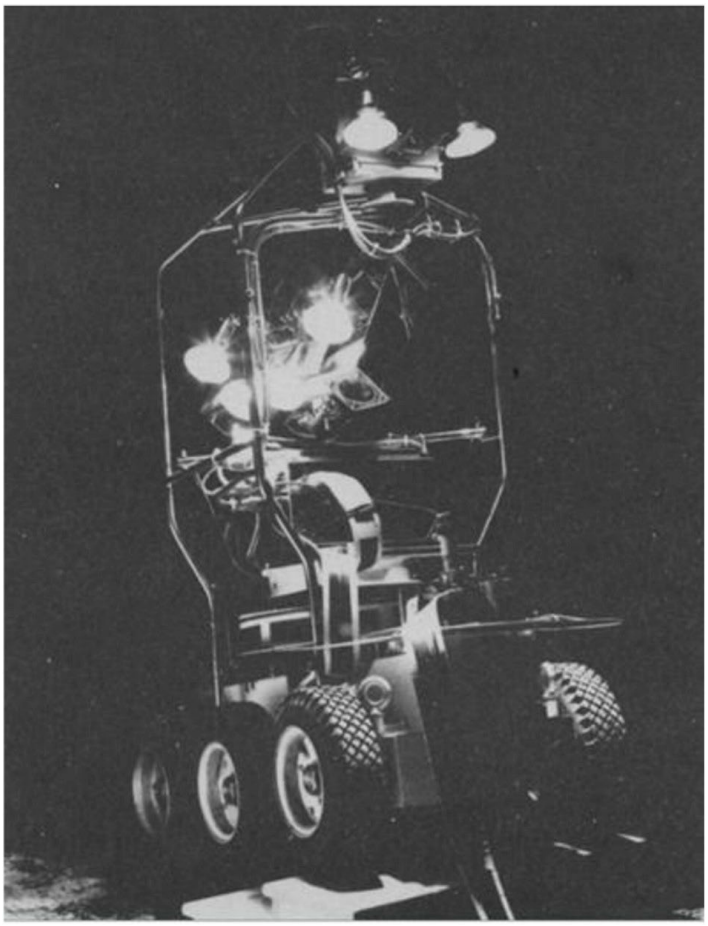


Figure 17. ROVER system deployed to Three Mile Island (Gelhaus & Roman, 1990).

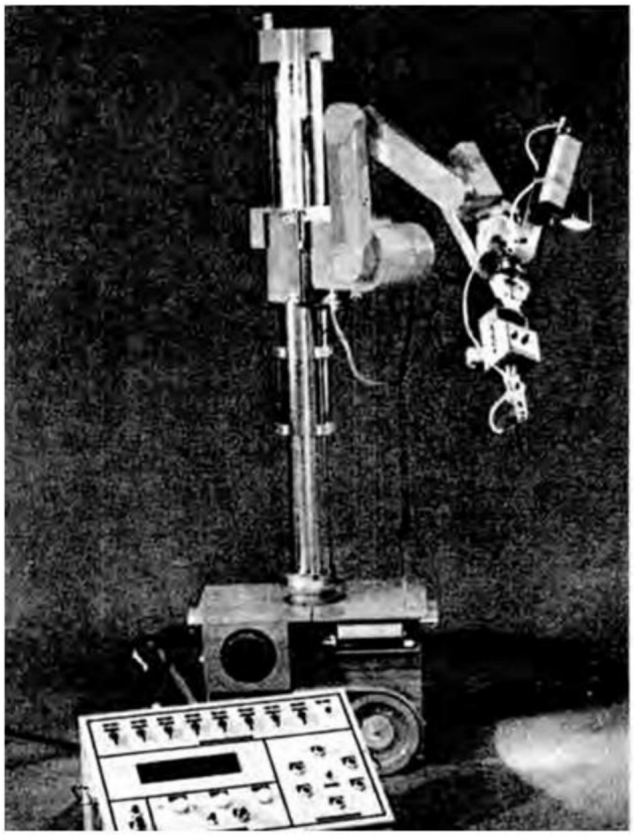


Figure 18. LOUIE I system deployed to Three Mile Island (Reilly et al., 1985).

When the Chernobyl accident occurred in 1986, robotics were part of the recovery and monitoring. Two teleoperated robots (STR-1) were deployed soon after the accident to clean debris off the roof, but their electronics were quickly disabled by the high radiation field. The cleanup had to be performed by humans in short shifts due to the high radiation exposure. To control the spread of radiation, a sarcophagus was constructed around Chernobyl Unit 4. As the sarcophagus aged, damage was observed on the structure, and robotics was the solution of choice for inspection. One platform used for this purpose was the Pioneer (Figure 19). To assist in radiation hardening, the Pioneer was tethered (some of the core components could be located outside via the tether) and

lead shielding was used around electronic components mounted on the platform (Maimone et al., 1998).

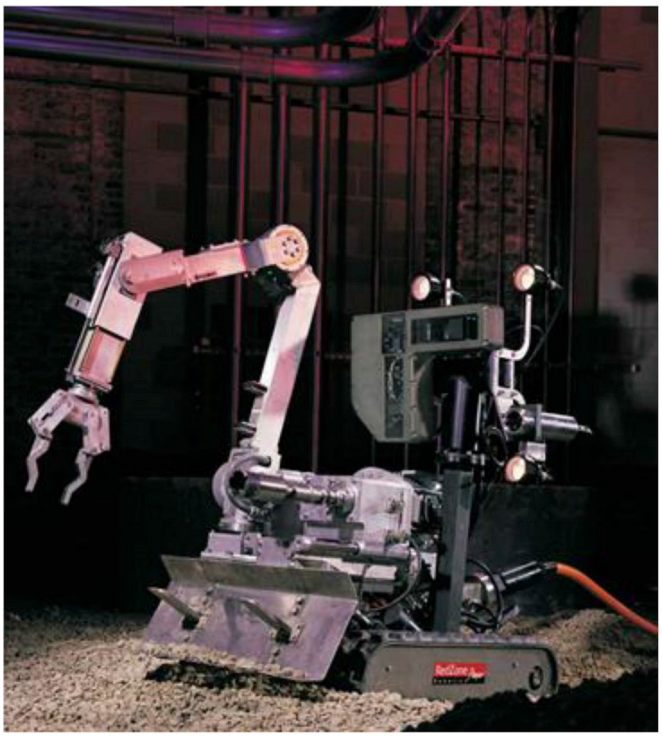


Figure 19. Pioneer system deployed to Chernobyl (Tsitsimpelis et al., 2019).

After TMI and Chernobyl, research expanded on the use of robots in radiological environments for cleaning, inspection, repair, and object retrieval. As of 1991, 44 nuclear power companies had implemented robotics to reduce exposure for human operators, but limitations in navigation, radiation hardening, and failure recovery limited the performance of robotics (Roman, 1991). In 1999, a criticality accident occurred in a fuel processing unit at Tokaimura in Japan. In response to this accident, a flurry of emergency response robots

were developed in Japan. One example is the series of RESQ (Remote Surveillance Squads) robots intended to survey contaminated locations (Figure 20).

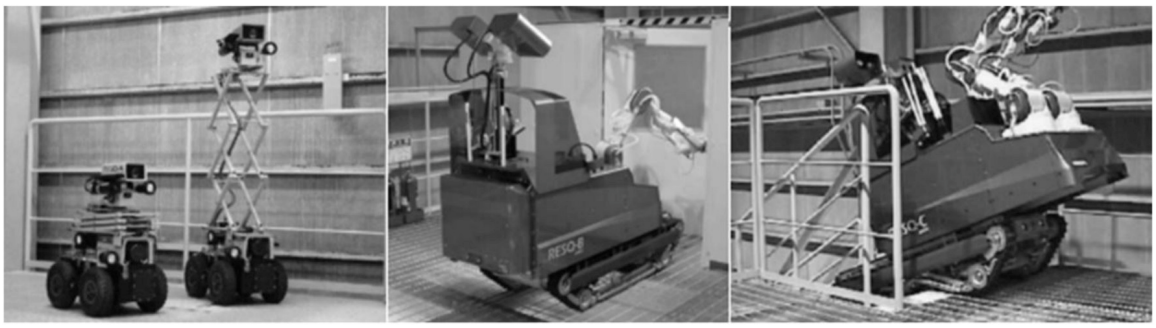


Figure 20. RESQ robots for nuclear accident information collection (Kobayashi, Miyajima, & Yanagihara, 2002).

One might think that with the technological advances over the past several decades and the focus on developing robotics for nuclear accident investigation, people would be ready to deploy robotics in a nuclear emergency scenario. However, this appears to be a lesson that has not been committed to memory. In the Fukushima Daiichi accident, the first robot in the facility was an American-made iRobot PackBot (Figure 21). Because the emergency situation necessitated creative problem solving, the PackBot had a radiation meter strapped to it with a camera pointed at the display to take readings (Figure 22).

Figure 21. iRobot PackBot operating at Fukushima (TEPCO, 2019).

Figure 22. Close-up view of radiation meter on PackBot at Fukushima (TEPCO, 2019).

A study in 2012 determined that robots that had been developed since the 1999 Tokaimura accident were not suitable for immediate deployment to Fukushima because they were not fit for the environment (either because of physical constraints or maintenance issues) or because too many modifications needed to be performed to make them usable for the situation. Some of the previous robots were able to be modified as the recovery went on, and one of the RESQ robots was altered (Figure 23) to perform gamma-ray imaging in the reactor buildings (Kawatsuma, Fukushima, & Okada, 2012). Many other systems have since been deployed for Fukushima, including, but not limited to, the Quince systems, Four-leg Walking Robot, and PMORPH (Figure 24).

Figure 23. JAEA-3 system for gamma imaging that was created from an earlier RESQ robot (Kawatsuma et al., 2012).

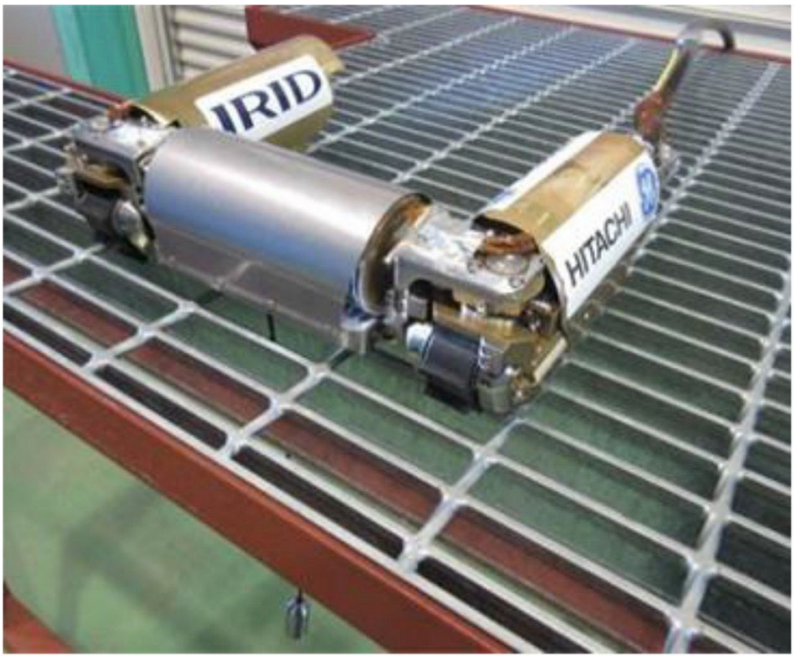


Figure 24. PMORPH robot to inspect the Unit 1 primary containment vessel at Fukushima (TEPCO, 2019).

Not every robot intended for a nuclear environment is intended for emergency response activities. Cortez, Tanner, and Lumia (2009) mounted sodium iodine scintillators on small Khepera II mobile robots to search for radiation (Figure 25). The work of Cortez et al. is significant because it includes counting statistics to analyze the measurement being taken. Incorporating the counting statistics (see Chapter 3) enabled two search strategies: a gradient-based Bayesian method to search toward the areas of highest uncertainty and a sequential-based Bayesian method that travels predetermined paths but time spent in certain sections varies with uncertainty.

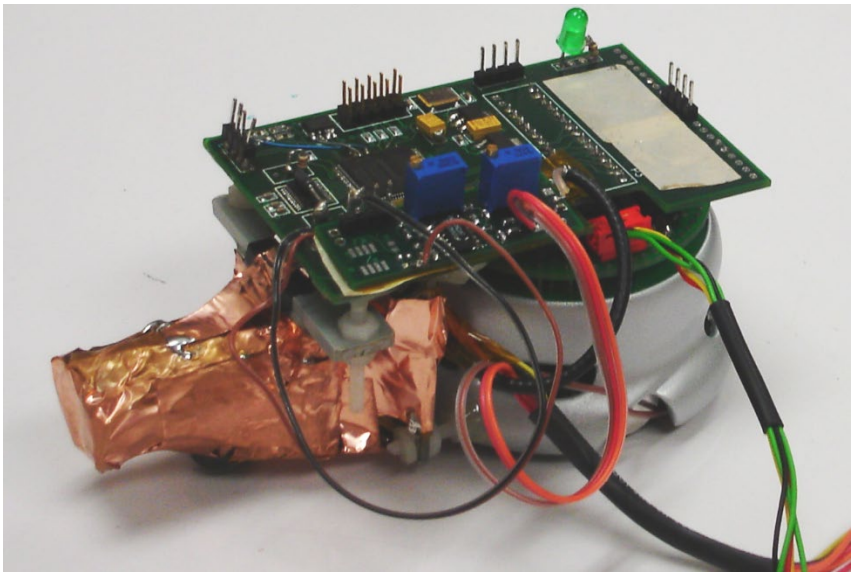


Figure 25. Khepera II mobile robot for radiation mapping (Cortez et al., 2009).

Ravishankar et al. (2013) created a mobile robot to carry instrumentation to map dose rates around various locations of beam lines off a cyclotron (Figure 26). This experiment was notable for the focus on neutron mapping (and gamma mapping), though the neutrons were of high energy. The mobile robot was controlled wirelessly and remotely so as to prevent exposure of human operators.

Figure 26. Mobile robot to map neutron and gamma dose rates for cyclotron beam line (Ravishankar et al., 2013).

Many other papers describing simple mapping of radiation by incorporating detectors with robots have been produced. An example of a gamma detector incorporated with a mobile platform similar to one of the ones used in this work can be seen in von Frankenberg, McDougall, Nokleby, and Waller (2012) (Figure 27).

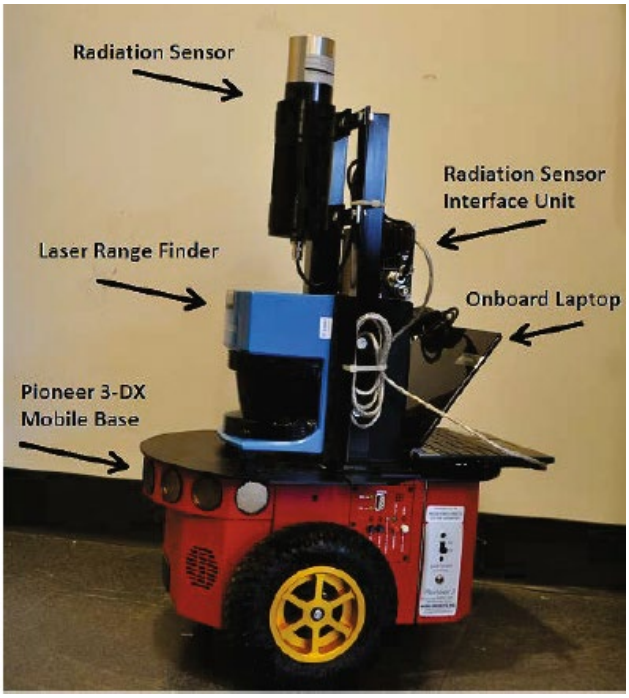


Figure 27. RadBot gamma mapping system on a Pioneer 3-DX base (von Frankenberg et al., 2012).

It is worthwhile to mention that radiation mapping has been incorporated with other modes of robotic locomotion such as unmanned aerial vehicles and submersible robots, but the particular challenges those applications face are not within the scope of this work (that is not to say that lessons cannot be learned from them, but rather that they are not so directly applicable to the problem of directional neutron surveying on a mobile ground robot). Table 1 presents a summary of some of the mobile platforms mentioned in this section and similar ones.

Table 1. Summary of mobile platforms used in radiological inspections (adapted from Tsitsimpelis et al., 2019).

Robot	Operating Areas	Communication	Locomotion	Terrain	Year
RRV-1	Three Mile Island, reactor basement	Tether	Six wheels	Water 305 mm/35° slopes	1984
LOUIE I	Three Mile Island, auxiliary buildings and cubicles	Tether	Two tracks	Rough	1984
KLAN	Chernobyl, sarcophagus	Wireless	Tracks	Unavailable	1986
MACS	Chernobyl, Unit 4 shelter	Wireless	Wheels	Flat	1995
RCS	Chernobyl, outdoor areas	Wireless	Wheels	Rough	1996
NOMAD	Chernobyl, Unit 4 shelter	Wireless	Wheels	Rough	1996
Pioneer	Chernobyl, sarcophagus	Tether	Tracks	45° slopes	1997
RESQ-A	Post-Tokaimura, mock-up reactor areas	Wireless/Tether	Four wheels	Slight inclines/small obstacles	2001
RESQ-B/C	Post-Tokaimura, mock-up reactor areas	Wireless/Tether	Tracks	Stairs	2001
SMERT-M	Post-Tokaimura, mock-up reactor areas	Wireless/Tether	Tracks	40° slopes	2002
SMERT-K	Post-Tokaimura, mock-up reactor areas	Wireless/Tether	Wheels	Slight inclines	2002
PackBot	Fukushima, Units floor inspection	Wireless	Tracks	All terrain	2011
JAEA-3	Fukushima, Unit 2 gamma imaging	Tether	Four wheels	Slight inclines/small obstacles	2011
Quince	Fukushima, Units 2 & 3 floors/gas ducts	Wireless/Tether	Six tracks	Rough/60° slopes	2011
Survey-runner	Fukushima, Torus Room	Tether	Four tracks	Obstacles 235 mm/45° slopes	2012
Tele-runner	Fukushima, suppression chamber	Unavailable	Tracks	Slight inclines	2015
Frigoma	Fukushima, near primary containment vessel (PCV)	Wireless	Four tracks	Obstacles 430 mm/45° slopes	2012
Rosemary	Fukushima, 1–3 floors, all units	Wireless	Four tracks	Rough/60° slopes	2013
Sakura	Fukushima, 1–3 floors, all units	Tether	Six tracks	Rough/60° slopes	2013
Kanicrane	Fukushima, 1st floor, all units	Tether	Two tracks		2014
PMORPH 1	Fukushima, PCV Unit 1	Tether	Tracks	Narrow/grating surfaces	2015
PMORPH 2	Fukushima, PCV Unit 1	Tether	Tracks	Narrow/grating surfaces	2016
SCORPION	Fukushima, Unit 2	Tether	Tracks	Narrow/grating surfaces	2016
SIMON	Savannah River, floors	Wireless	Three wheels	Flat	1990
Inspection crawler 1	Savannah River, H-Canyon	Tether	Two tracks	Standing water/slight curbs	2003
Inspection crawler 2	Savannah River, H-Canyon	Tether	Two tracks	Standing water/slight curbs	2009
Inspection crawler	Savannah River, H-Canyon	Tether	Four wheels	Standing water/slight curbs	2014
Recovery crawler	Savannah River, H-Canyon	Tether	Four wheels	Standing water/slight curbs	2015
SURVEYOR	Drum waste storage	Wireless	Two tracks	Water 152 mm/obstacles 229 mm	1985
ROCOMP	Multi-purpose	Wireless	Two tracks	Stairs	1986
SURBOT	Multi-purpose		Three wheels	Water 76 mm/obstacles 38 mm	1985
HERMIES III	Waste storage surfaces	Tether	Wheels	Flat	1989
Kaerot/m2	Pressure tubes in PHWR	Tether	Four tracks	Stairs	2003
RICA	Fuel retreatment/Waste storage	Tether	Two tracks	Rough/slight slopes/obstacles	2016
MOTHERSHIP/modular snake	Waste storage	Tether	Track modules	Rough/narrow	2016

2.3 Robot Control Methodology

The previous section showed many examples of robots used in nuclear environments, but they are overall lacking in full autonomy because they must be directly controlled by human operators via wireless or tethered communications. However, the levels of autonomy that can be implemented in robotic systems are varied rather than all or nothing. When controlling a robotic arm, for example, each individual joint can be directly controlled manually by a human operator. Alternately, the operator can control the position of the end-effector in space while the robot calculates the kinematic solution for the individual joints to reach that position. In both arm control cases the human remains in control of the high level decision on where to move. Various work has been done historically to define the approach to robot control when human operators and robots are working together. Goodrich (2004) used a human cognition model to develop a list of principles to guide the design of *efficient* human-robot systems:

1. Implicitly switch interfaces and autonomy modes.
2. Let the robot use natural human cues.
3. Manipulate the world instead of the robot.
4. Manipulate the relationship between the robot and world.
5. Let people manipulate presented information.
6. Externalize memory.
7. Help people manage attention.
8. Learn.

Goodrich's design principles help to lay the groundwork for assessing various levels of autonomy because they describe what successful sharing of control between robot and operator could look like. These principles all aim to reduce the burden on the operator. Parasuraman, Sheridan, and Wickens (2000) proposed an autonomy framework that

evaluates systems based on human performance consequences and robotic reliability. Their framework applied automation on a continuum of levels from low to high in the function categories of information acquisition, information analysis, decision and action selection, and action implementation. Modeling of human robot interaction with varied levels of control is not a recent idea in robotics though increased computing power has increased the sophistication of tasks that can be automated. As undersea operations for exploration, inspection, construction, maintenance, salvage, and rescue progressed to increasing depths, the dangers also increased for human divers to perform these tasks. Growth in the reliability and performance of computers and electrical equipment in the 1970s drove increased interest in the use of teleoperated submersible machines. Sheridan and Verplank (1978) described ten levels of increasing autonomy for undersea teleoperation tasks ranging from full teleoperation to full autonomy with shared control between. Riley (1989) proposed a taxonomy to describe increasing levels of autonomy while using the term "mixed initiative" to describe the shared control between robot and operator. Bruemmer has published extensively on "mixed initiative control" as a paradigm for autonomy levels. Bruemmer et al. (2005) proposed four modes of robot control:

- *Tele mode* is full manual operation of the robot by the operator.
- *Safe mode* is similar to tele mode but a level of initiative is granted to the robot in order to avoid collisions.
- *Shared mode* is a dynamic allocation of roles and responsibilities where the robot can relieve the operator's burden via movement planning and execution while still accepting varying levels of operator intervention as needed.
- *Autonomous mode* is operation while the robot manages decisions and navigation.

Brabec et al. (2011) delineated "transitional levels of autonomy" that represent a scaled approach to define the interstitial modes of operation between full teleoperation and full autonomy for "contact tasks" that have the robot make physical contact to interact with its environment (Table 2). According to the transitional levels of autonomy, this work would function around level 5 because the operator can "quickly direct the system to complete tasks" while the robot is handling the responsibility for moving and avoiding collisions (though not all the levels cleanly apply to a system not currently seeking to contact or grasp objects in its environment). However, the transitional levels reinforce the important concept that the level of autonomy in a system is on a scale that can be increased over time to meet changing operational requirements and desires.

Table 2. Transitional levels of autonomy (Brabec et al., 2011).

Level	Teleoperation → Autonomy
1	Reduce or eliminate operator's need to manage the robot's internal configuration.
2	Reduce or eliminate the operator's responsibility for avoiding undesired contact with the environment.
3	Reduce or eliminate the operator's responsibility for moving the robot to locations of interest.
4	Reduce or eliminate the operator's responsibility for selecting a proper grasping configuration for retrieving selected objects.
5	Allow the operator to quickly direct the system to complete tasks that involve subtasks completed as directed in levels 3 & 4 (such as pick and place).
6	Reduce or eliminate the operator's responsibility to avoid threshold forces for contact tasks such as opening a door or lifting items exceeding the system's payload.
7	Reduce or eliminate the levels of detail that are necessary for the operator to communicate a task (or subtasks) to the robotic system.

Table 2 Continued

8	Integrate capability to complete tasks that require high levels of precision and/or the control of a specific force profile.
9	Reduce or eliminate the need for the operator to be in the loop for tasks that respond to non-operator, independent external events (i.e. timer on oven, low battery notification, etc.).
10	Based on prior tasks completed, the system anticipates future tasks to be completed based on historical use.

The nuclear industry tends to be risk-averse and new technologies must be rigorously proven before being fully adopted. Keeping human operators in the control loop helps to build trust in the system while humans remain in charge of any high level decisions (and can intervene in case of problems during operation), but reduces the burden on those operators by offloading movement and sensing to the robot. It is this shared mode of control that is meant by the term semiautonomous used in this work. A fully teleoperated system would be directly controlled by a human operator and have no decision-making power of its own. Conversely, a fully autonomous system operates without the aid of a human and has all the decision-making power. Semiautonomous refers to a system between these two extremes similar to the previously described autonomy frameworks. As trust is built up in a semiautonomous system, it could eventually transition into higher levels of autonomy.

2.4 SUMMARY

This chapter summarized some previous efforts to measure or control the direction of detected neutrons. The previous attempts presented to measure direction are typically unwieldy in size, need finely tuned controls to operate the detector, or are better suited for high energy neutrons than would be seen in a typical LANL SNM storage area. Using

collimators to control the direction of neutrons is a useful concept to apply to this work, but has been done on a much finer scale than would be needed to investigate containers of SNM. Additionally, many efforts using robotics to survey radiation are similar to what was originally envisioned with this work, but would not be immediately suitable. Robots intended for nuclear accident response are frequently teleoperated via tethered cables or wireless communications that would not be desirable or currently allowed at LANL. These post-accident robots have limited autonomy in their mobility, but some of this can be driven by the unknowns inherent in post-accident response. The limited availability of teleoperation at LANL drives any implemented system towards a shared level of control between robot and operator that has been previously modeled in various autonomy taxonomies from robotics. Further explanation of the LANL restrictions on the use of wireless and tethered communications, and resulting level of autonomy, follow in the next chapter. Robotic systems used to map radiation during regular operation of a facility are similar to this work, but have been either teleoperated, focused on other types of radiation, or have not tried to measure some directionality to the detected neutrons.

Chapter 3: Methodology

Historically, the inclusion or consideration of robotics within the LANL Plutonium Facilities (including PF-4) has been met with vociferous skepticism. Robotic systems are believed to be notorious for high maintenance requirements and increased probabilities of failure, requiring time-consuming work within the facility and often within gloveboxes. However, a primary driver for considering robotic systems is the reduction of worker exposure to ionizing radiation. PF-4 has considered and installed robotics in the past. For example, certain programs have installed the Robotic Integrated Packaging System (RIPS) (Scheer et al., 2002), and the machining lathe is also highly automated to the point of being sometimes called a "robotic lathe" (Brown, 2004). On the other end of the spectrum, PF-4 workers are keen to note the inherent beauty of simplistic systems like a rope-and-pulley transfer system in the foundry trunk lines.

The primary justifications for considering robotic systems, in general, in the plutonium facilities are thus proposed:

- Improved radiological safety (reduced worker dose and increased monitoring of stored materials)
- Improved industrial safety (reduction of worker injuries, ergonomic or otherwise)
- Increased throughput (automation to reduce process time)
- Increased quality (automation to reduce product variability)
- Increased security (reduction of material diversion risks)

The radiological risks are varied throughout the plutonium facilities, but the operations of storing and retrieving material usually have large dose rates because of the large amount of material in close proximity. The vault and other storage locations are thus prime locations to target reduction of worker dose. Eventually, a fully automated material

retrieval system would be beneficial for LANL; in the meantime, a survey robot would provide many of the same benefits and would build trust without putting it in direct contact with sensitive materials. A key component of managing the material is performing periodic inventory checks—a time-consuming task that increases the radiological dose received by those executing the task. Similarly, dose maps for areas of high radiation are performed infrequently (as mentioned in Chapter 1) but provide necessary information to help ensure worker safety. Automating such operations reduces hazards to workers while increasing the regularity of scans for more up-to-date information.

The goal of this work is to explore the implementation of a mobile robot that can perform directional radiation scans in areas where nuclear materials are stored. While the eventual goal is to develop a system that is capable of performing radiation surveys within a vault, the final integration of the robotic system into PF-4 is outside the scope of this dissertation. The reasons for this are twofold: Numerous other safety and regulatory approvals would be needed for system installation, which are not directly related to the initial design of this system (for example, criticality safety or safety basis review), and initially keeping a robot out of a vault makes it far easier to share data without worrying about classification issues. That being said, many other operational challenges had to be addressed to operate even in areas without significant quantities of nuclear material. The high-level experimental plan was to model and then fabricate a neutron shield to enclose a neutron detector that contains an aperture to bias the directionality of detection. Following this, the peripherals were then to be successfully integrated with a robotic system, and that system needed to demonstrate an ability to maneuver around a room and take radiation measurements (Figure 28). This chapter explores the equipment, software, and strategy used in pursuit of this goal; the results are detailed in the following chapter.

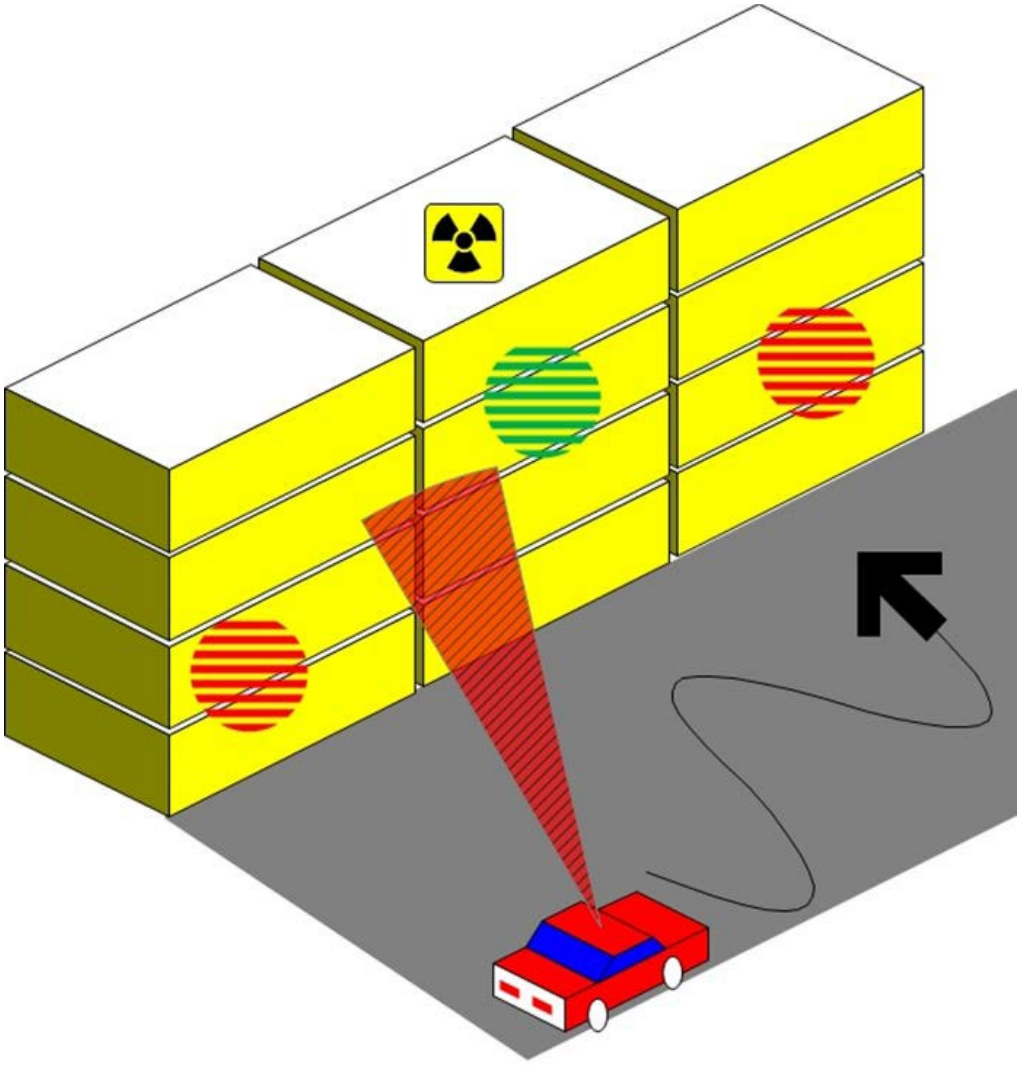


Figure 28. Illustrative representation of the final system able to drive around a room and scan locations for radiation.

3.1 Equipment and Software

This section details the equipment, software, and facilities used in this work.

3.1.1 Neutron Detector

The cornerstone piece of equipment is the neutron detector. A Bridgeport Instruments neutron detector (R2D-NT-1X12) was purchased for this work (Figure 29). This detector uses boron-10 to detect neutrons via scintillation. The detector is approximately 20.25 inches long with a diameter of about 2 inches but can also be manufactured in custom shapes and sizes. The active volume is approximately 12 inches long by 2 inches wide and tall.

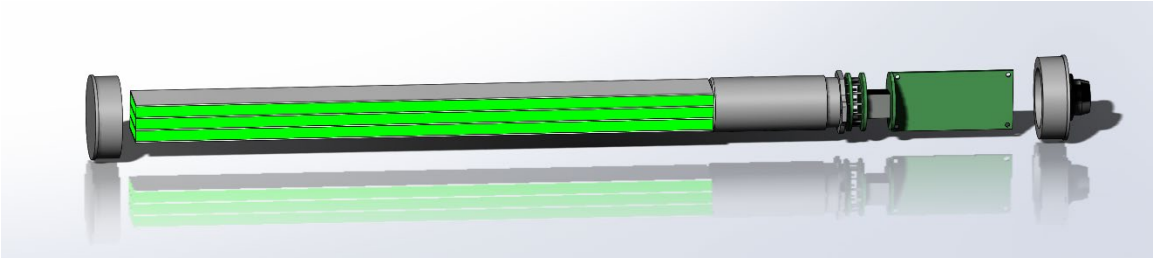


Figure 29. Internal view of neutron detector.

Neutrons are one of the more difficult types of radiation to detect, largely due to their neutral charge. Because many types of radiation ionize atoms, thereby making it difficult to distinguish the type of particles producing the ionization, neutron detectors operate by detecting the products of secondary reactions, such as absorption and scattering. The Bridgeport Instruments neutron detector operates based on an absorption reaction involving boron-10. Thermal neutrons have a large cross section (3,840 barns) for the (n, α) reaction:

$$^{10}_{5}\mathrm{B} + {}^{1}_{0}\mathrm{n} \rightarrow {}^{4}_{2}\mathrm{He} + {}^{7}_{3}\mathrm{Li} + 2.79\ \mathrm{MeV}$$

Roughly 94% of the time this reaction leaves the lithium nucleus in an excited state, which will then emit a gamma ray with energy of 0.48 MeV (Knoll, 2000). The Bridgeport Instruments detector contains a mixture of a boron-10 containing compound with ZnS(Ag) scintillator material. A thin layer of this mixture is deposited around poly(methyl methacrylate) rods (PMMA is also commonly known as Lucite). This material has a dual purpose: The PMMA serves as a light guide to channel light from the scintillator to photomultiplier tubes for counting in the 4,000-channel multichannel analyzer (MCA), and it acts as an internal moderator to increase the sensitivity of the detector to fast neutrons (Bridgeport Instruments, LLC, 2012).

The boron-10 cross section has a $1/v$ dependence over a wide range of energies (Figure 30).

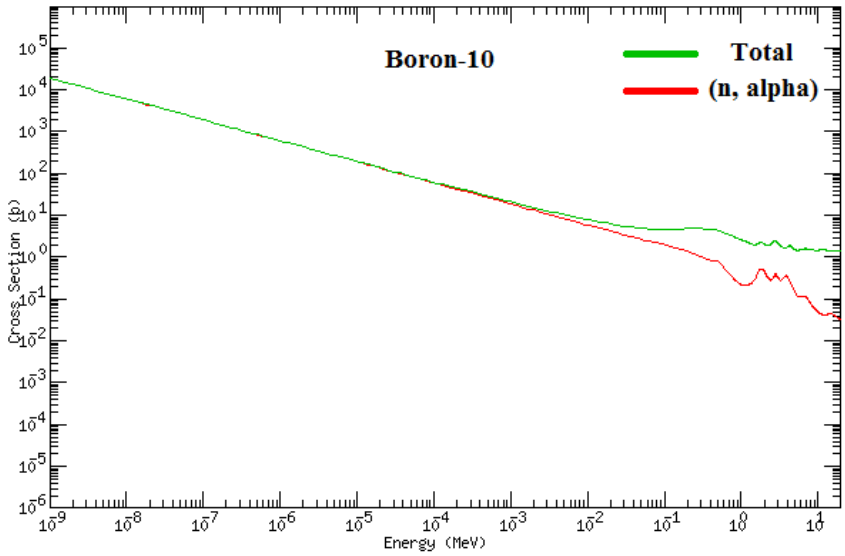


Figure 30. Total and (n, α) cross sections for boron-10 (KAERI, n.d.).

The detector has a base efficiency of about 4.8% on unmoderated Cf-252 neutrons. Moderating the source of the Cf-252 neutrons was able to boost this efficiency to 10.2% in company testing. Deadtime is 5 μs for neutrons and 0.6 μs for rejected gamma rays, though the detector's firmware can compensate for dead time automatically. The detector has strong gamma ray rejection that can be made stronger at the expense of neutron efficiency (Bridgeport Instruments, LLC, 2012).

3.1.2 MCNP

Monte Carlo N-Particle (MCNP) is a computer code developed by LANL for three-dimensional radiation transport simulations. Its LANL history dates back to the 1940s, when legendary mathematician John von Neumann proposed using the Monte Carlo statistical method of repeated random sampling for neutron diffusion and multiplication problems. A Monte Carlo method is an algorithm to obtain numerical results from complex phenomena using probabilistic sampling. For radiation transport, MCNP simulates individual particles and records their average behavior as governed by their interactions with materials (T. Goorley, et al., 2012). The most recent, major version releases are MCNP6.1 (2012) and MCNP6.2 (2017) (LANL – MCNP, 2018). MCNP currently supports a wide variety of particles and energies for simulation (Figure 31).

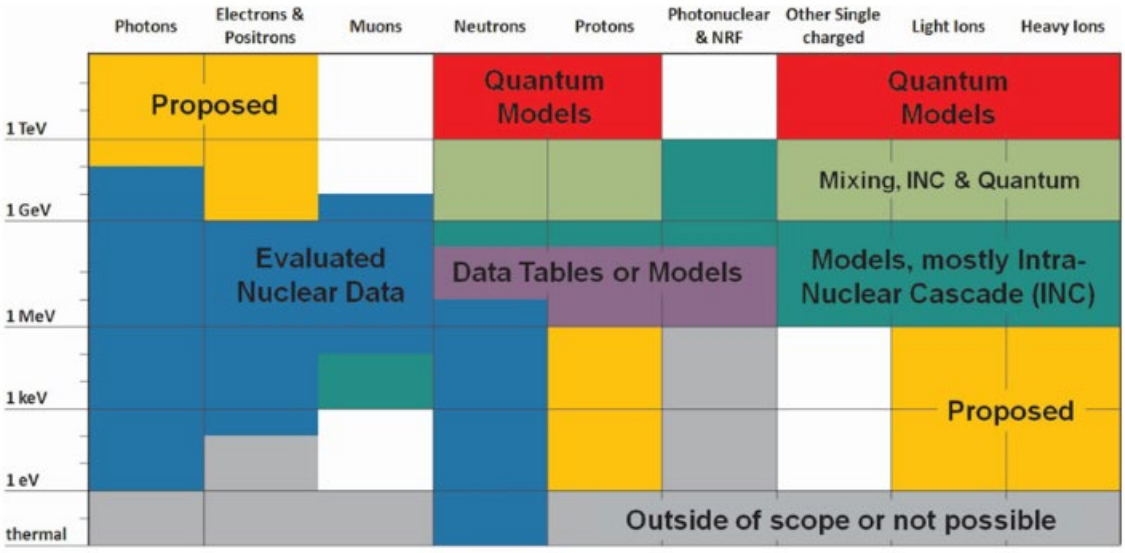


Figure 31. Tabular representation of MCNP6 particle types, energy ranges, and interaction physics (T. Goorley et al., 2012).

MCNP uses data from the evaluated nuclear data files (ENDF/B) for neutron cross sections. The ENDF project is an international collaborative effort between national laboratories and universities to evaluate, test, and manage cross section data (T. Goorley et al., 2012). MCNP6 was released during the ENDF/B-VII.1 version release (2011), although there is a new release of ENDF/B-VIII.0 (2018) ("Evaluated Nuclear Data File (ENDF)," 2018).

In a typical MCNP setup, a problem is defined by modeling the environment with a constructive solid geometry (CSG) method entered into a text file input. In CSG, simple primitive shapes such as spheres and cylinders are combined with Boolean set operators to union, intersect, and difference the primitives to create more complex geometry. However, in truly complex geometries, adequately representing the shapes can be a challenge using CSG. MCNP6 introduced the ability to more accurately and easily build complex models

by using an unstructured mesh. Unstructured mesh (UM) geometry provides "the capability to track and tally neutrons and photons on an unstructured mesh embedded as a mesh universe with MCNP's CSG" (T. Goorley et al., 2012). Complex models can be built in a 3-D computer-aided design (CAD) modeling program and then meshed using software such as Abaqus (used for finite element analysis) or Attila (a discrete ordinates transport code that can be used solely for meshing). The use of unstructured mesh was explored and evaluated for this work. Although there are advantages to using UM, it can significantly increase processing time. Due to the relatively simple geometries of the neutron detector, it was determined that the advanced capabilities of UM were not needed to accurately simulate this system.

3.1.3 Mobile Platforms

This work includes the use of two different mobile platforms. One platform, the Adept Pioneer LX, was used extensively for initial testing and the work of obtaining operational permission within one location of the main plutonium facility. Another platform, the Vector, later became available as part of another project of the author's. When it became apparent that the Vector platform would allow the incorporation of actual nuclear material for testing (which would not have been permissible without extensive review in the plutonium facility with the Pioneer platform), it was decided to incorporate the detector with the Vector system. This circumstance was not directly due to issues relating to the robots though the Vector does have more advanced sensors and greater computing power than the Pioneer. Creating an equivalent setup using nuclear material with the Pioneer would have required an amalgamation of extreme time investment and operational difficulties. Getting a novel radiological activity permitted in the plutonium facility while

securing the use of material and certified material handlers or removing the Pioneer from the facility would have required extensive time and effort and may not ever have been allowed. The availability of a separate facility with sealed sources and a willingness to allow testing is what drove the transition to the Vector. However, the preliminary work with the Pioneer platform was instrumental in the development of this application's semiautonomous robot control methodology and gaining approvals to procure and operate the integrated Vector system.

3.1.3.1 Adept Pioneer LX

The first platform identified and procured to meet the goals of a semiautonomous radiation detection and mapping robot was an Adept Pioneer LX (Figure 32). This mobile platform uses sonar, a laser rangefinder (with 270° field of view), and physical contact sensors ("bumper panels") to enable it to map its surroundings in real time and locate itself within them. It can carry up to 60 kg over indoor surfaces at speeds up to 2 m/s. It is designed for up to 13 hours of continuous operations on a full charge with a battery lifetime of seven years. The system itself weighs 60 kg and measures 19.7 inches wide, 27 inches long, and 17.6 inches high. It features an onboard Linux computer, USB ports, A/V ports, and onboard power supply for attached peripherals. Should the platform lose power or malfunction, it can be stopped by the emergency stop button, it can be placed into a direct teleoperation mode, or it can easily be physically pushed via a brake release button. This system supports the use of ROS (Robot Operating System) but came preinstalled with its custom operating software. Wi-Fi networking capability would typically be included on this platform, but it was removed at LANL's request to comply with LANL security requirements.

Figure 32. The Adept Pioneer LX mobile robotic platform (Adept MobileRobotics, 2013).

A microcontroller is responsible for controlling the platform's mobility by integrating data from the system's encoders and gyroscope and responding to signals from the emergency stop, bumper panels, and optional joystick control. Two SDKs are included with this particular system to aid in autonomous navigation. The Advanced Robotics Interface for Applications (ARIA) is a framework for controlling and receiving data from the platform. ARNL is a laser navigation and localization library to incorporate the data from the laser scanner. Also included are various software applications to assist in the operation of the robot. The platform can be simulated using MobileSim; MobileEyes acts as an interface to the ARNL library to visualize and operate the platform (Figure 33); and Mapper3 can be used to generate and edit local maps created by scanning with the laser (Adept MobileRobotics, 2013).

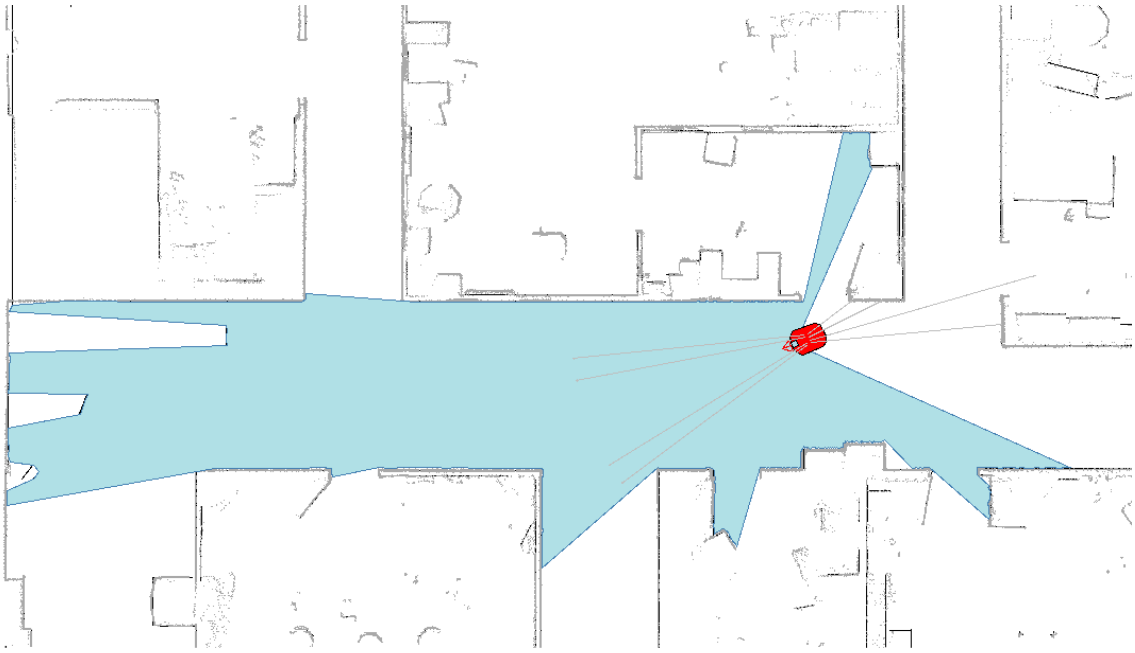


Figure 33. Pioneer MobileEyes software showing the scanned map of uncleared location at LANL (created with Mapper3) and the range of the laser scanner in blue.

3.1.3.2 Waypoint Robotics Vector

Another mobile platform, the Vector model from Waypoint Robotics, was available later in the timeline of this project (Figure 34). As previously mentioned, incorporating this platform enabled the use of sealed radioactive sources to collect actual radiation measurements and verify the function of the neutron detector shielding. The Vector is intended as a warehouse support platform. It has a ROS-native system architecture (Kinetic Kame distribution release). The platform is rated for a maximum payload of 300 lb and can achieve speeds up to 2.0 m/s. There are two safety Light Detection and Ranging (LiDAR) laser scanners—on the front and back kitty-corners—that activate a safety stop to avoid collision if an object is detected too close to the robotic base. The system can be

immediately stopped by engaging the red emergency stop button. During path planning and simulation, a cost map is used to model the base (with additional safety thresholds extending beyond the physical size of the base) to prevent collisions. Path planning was implemented using an elastic band method. An additional LiDAR unit (Velodyne VLP-16) is mounted near the center to dynamically visualize the environment and detect obstacles during platform operation. The system has excellent mobility due to its mecanum wheels, which allow omnidirectional movement. This platform normally comes standard with Wi-Fi networking capability, but due to current security requirements, the manufacturer removed all wireless networking at LANL's request, as with the Pioneer LX platform.

Figure 34. Waypoint Robotics Vector mobile platform at LANL.

3.1.4 ROS

Robot Operating System (ROS) is an open-source messaging framework designed to support robotic systems. It provides “hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management” (ROS.org, 2018). ROS was designed to be lightweight, modular, and programming language–independent (it has been implemented in C++, Python, Octave, and Lisp, with testing done in Java and Lua). A typical structure to a ROS implementation will be a network of many nodes that act like processes to perform some sort of computation. Communication between nodes is achieved through messages with predefined data structures. Messages can be sent as either topics or services. A topic is defined with a name (a string data type), and other nodes can subscribe to receive information published to that topic. A service is a request-and-response paradigm. Services are also defined with string names, and the request and response data types are defined (Quigley et al., 2009). ROS has grown since its introduction in 2009 and currently supports a wide range of platforms and sensors used in robotics.

3.1.5 Elastic Band Planning

The Vector platform currently uses an elastic band planner for motion control. Elastic bands are a framework to incorporate path planning with sensor feedback to create a deformable path that avoids collisions as a robot moves through its environment. After a given path is calculated by a planner, the path can be smoothed via an internal contraction force and an external repulsion force. These forces are what inspires the name for the method, because it approximates the tension seen in a stretched-out elastic or rubber band.

If an obstacle is found in the path, the band is repelled by it and the path deforms to avoid collision (Figure 35).

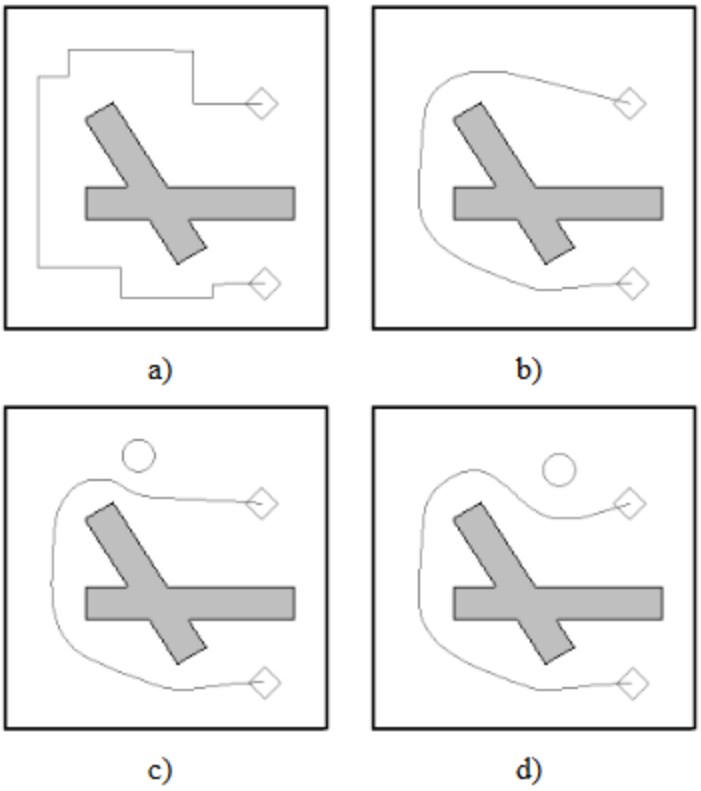


Figure 35. A) A path generated by a planner; B) applying both an internal contraction force and an external repulsion force; C) a new moving obstacle is introduced; D) the path is deformed to avoid the obstacle (Quinlan & Khatib, 1993).

The ROS implementation of the elastic band method computes a path using the costmap for the robot (Connette, Marthi, & Khandelwal, 2018). The costmap is a representation of the environment that uses an occupancy grid to represent obstacles. It is related to the footprint of the robot in both 2-D and 3-D (for visualization purposes the 3-

D can be projected down onto the 2-D map). The “cost” of a location, or cell, is inflated by propagating out the cost values of occupied areas and the values decrease with distance (Figure 36).

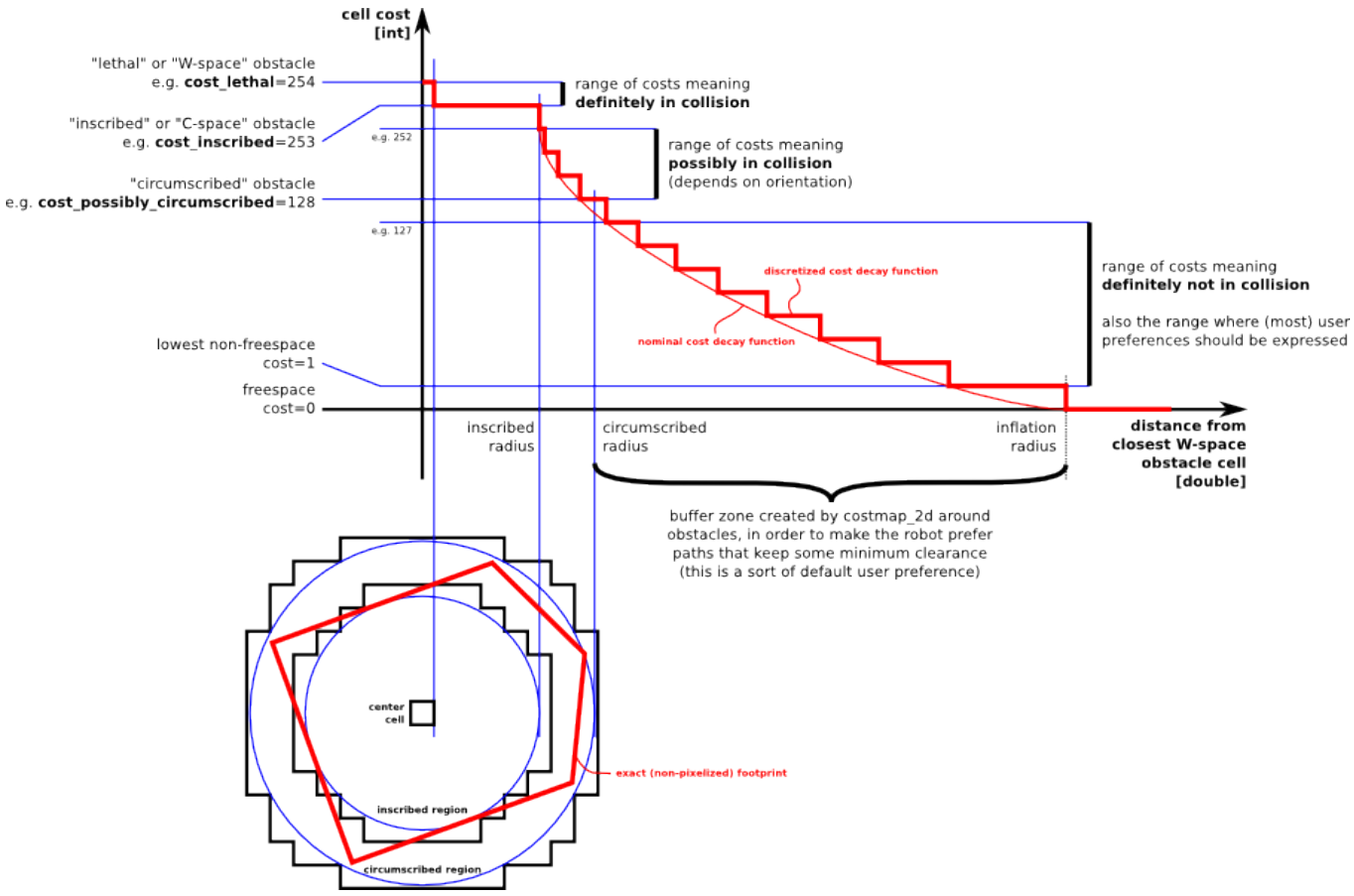


Figure 36. Inflation of costmap values as related to robot footprint areas (Marder-Eppstein, Lu, & Hershberger, 2018).

ROS by default uses five costmap categories in the 2-D representation (Marder-Eppstein, Lu, & Hershberger, 2018):

- Lethal: There is an obstacle in this cell, and if the robot is here, then it will be in collision.

- Inscribed: This is near an obstacle, and the robot's defined radius is such that if the robot is here, then it will be in collision.
- Possibly circumscribed: Similar to "Inscribed," but depending on the robot's orientation, it may not be in collision (for example, a robot that is longer than it is wide may be in collision facing one way but not in another).
- Freespace: There is no collision possible in this space and the robot can move freely.
- Unknown: Not enough information about this cell to determine the cost.

A visual representation of the costmap of the Vector platform as well as a simulation of the actual environment are presented in Figure 37 and Figure 38.

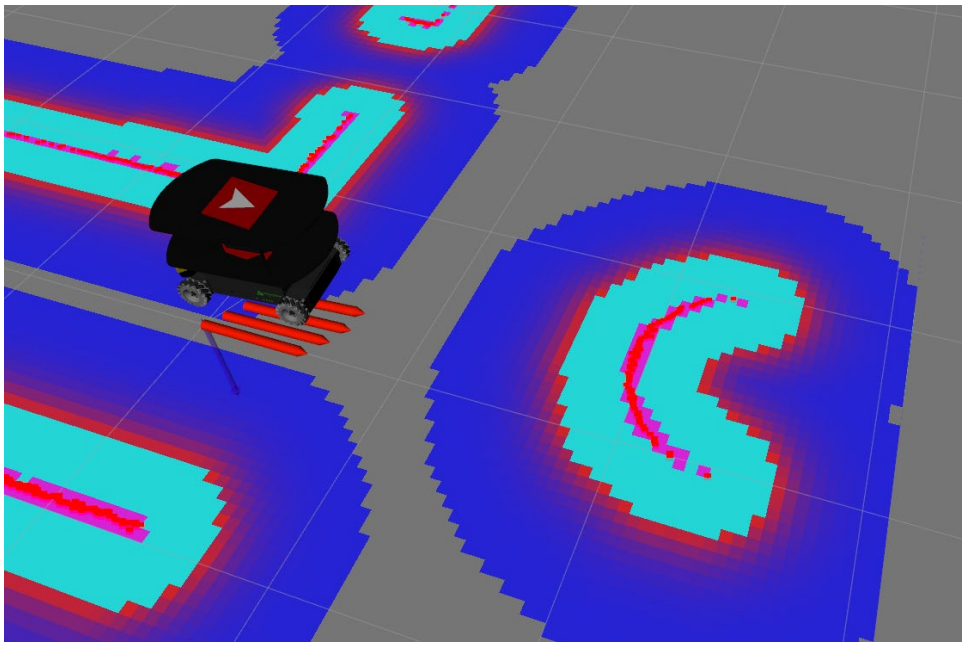


Figure 37. 2-D costmap visualization for Vector robot.

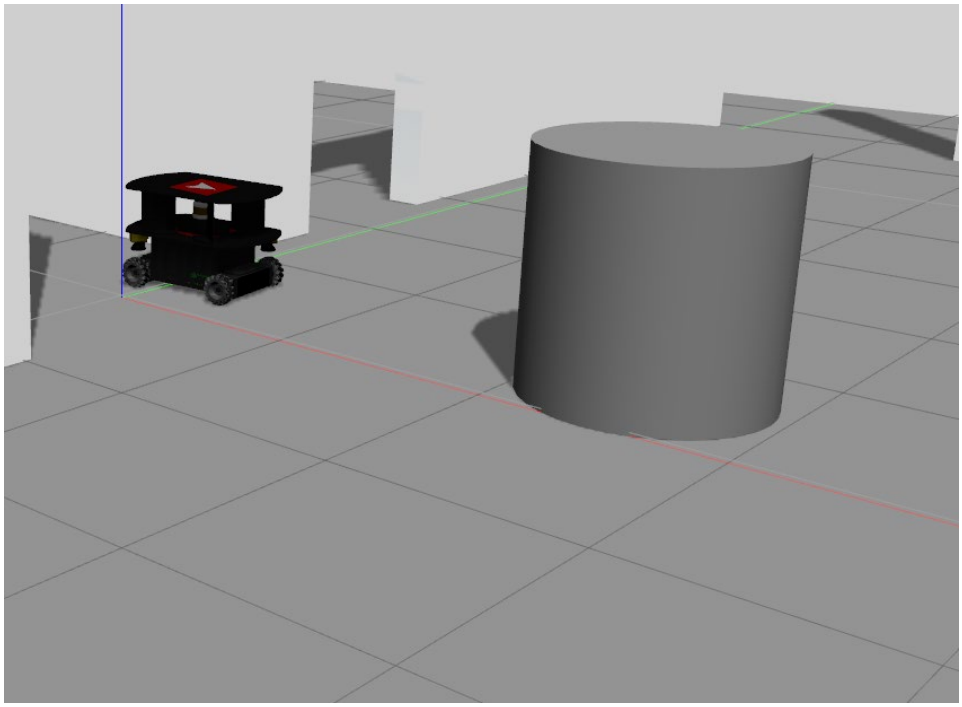


Figure 38. Visual simulation of the environment that correlates to the costmap in the previous figure.

3.1.6 Vector Peripherals

While the original emphasis of this work was radiological, other components are integrated with the Vector system to augment inspection of stored nuclear materials. Mounted on top of the base platform is a Universal Robots UR5e arm (Figure 39) and its associated controller box. This model arm has a reach of 850 mm and a payload of 11 lb. The arm itself weighs roughly 45 lb. The UR5e arm connects to the local Ethernet network on the Vector platform. This particular type of arm is called a collaborative robot or "cobot," which means it was designed to safely work in human-centric spaces. The system can be immediately stopped by engaging the red emergency stop button on the teach pendant or when detecting a collision (Universal Robots, 2019). Having a "cobot" versus a high-power arm—as one would find in the automotive industry—offers some safety benefits. DOE nuclear facilities tend to be older buildings designed with human workers in mind. Thus, having an arm that fits nicely into this paradigm requires minimal facility alteration for installation as compared with erecting safety guarding and barriers around automotive-style arms.

Figure 39. Universal Robots UR5e arm (Universal Robots, 2018).

Two cameras are mounted on the end-effector of the arm with a custom machined aluminum plate. The Optris Xi 400 is an infrared camera with a motorized focus and a framerate of 80 Hz (Figure 40). It has a compact and ruggedized body that can be further augmented for harsh environments if desired. USB connection provides easy incorporation with the robotic system (Optris, 2015). The other camera is a grayscale FLIR Grasshopper 3 visual camera (Figure 41). This camera also has USB connectivity. It is an 18FPS charged-coupled device (CCD) camera sensor with a 2,016 x 2,016 resolution (FLIR, 2019). Attached to the visual camera is a Tamron 16mm 1/1.8 inch C mount lens (model LENS-160T4C). The cameras as mounted are shown in Figure 42.

Figure 40. Optris thermal camera (Optris, 2015).

Figure 41. FLIR visual camera (FLIR, 2019).

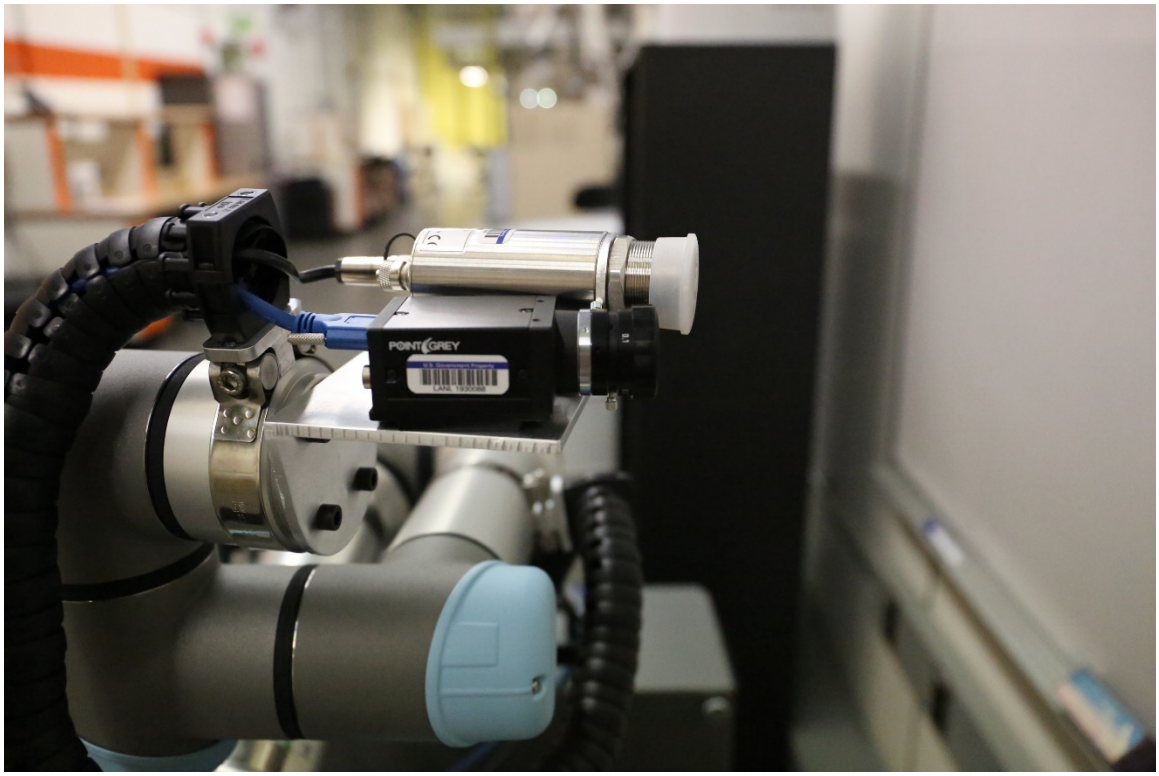


Figure 42. Visual and thermal cameras bolted to arm end-effector.

All the peripherals as currently integrated are shown in Figure 43. The arm is powered via an additional battery (12 volts absorbent glass mat lead acid with a capacity of 100 amp hours) and DC-AC inverter (1,000 watts) connecting to its control box on the back of the platform. The arm itself is mounted higher up to increase useable workspace area.

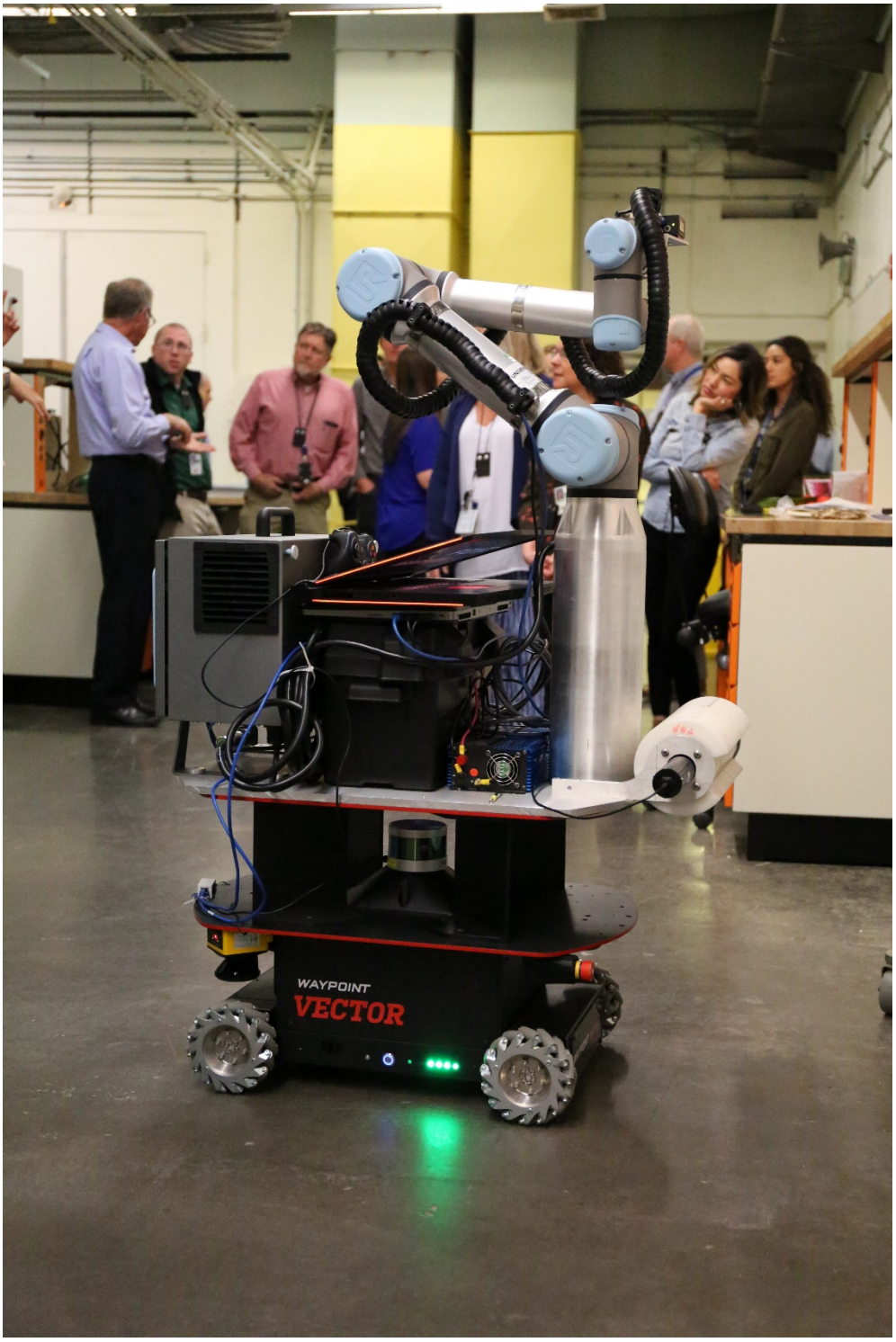


Figure 43. Vector platform integrated with arm and sensors.

3.1.7 Testing Facilities

The equipment listed in this section was tested across three facilities at LANL. The Pioneer LX was first installed in an uncleared building at TA-55, the Radiological Laboratory Utility Office Building (RLUOB). As will be detailed in the next chapter, testing and demonstrations had to be performed in this location in order to gain approval to relocate the mobile platform into a restricted space. Once approval was granted, the Pioneer was transported into a non-radiological room in PF-4 (Figure 44).

Figure 44. Pioneer LX mobile platform located in PF-4.

Based on demonstrations with the Pioneer LX, the Vector mobile platform was able to be placed into Building 27 at Technical Area 35 (TA-35) at LANL (Figure 45 and Figure 46). This building was previously designed to house a reactor but was never used for this purpose. Located in this building are hot cells, gloveboxes, and sealed sources. Sealed sources are allowed to be used in some areas of this building, and that was the motivation for selecting this area for testing of the platform and neutron detector.

Figure 45. View of testing hallway at TA-35.

Figure 46. Alternate view of testing hallway at TA-35.

3.2 Strategy and Theory

This section details some the operational constraints and considerations for running robots at LANL. Additionally, it goes through some of the mathematical, material, and statistical concerns for creating an artificially apertured neutron detector.

3.2.1 Operational Constraints

A challenge for this work is to operate within the security parameters for the area of operation. For a facility like PF-4, this can be somewhat limiting, but not impossible.

No wireless communications are currently permitted within PF-4 due to security restrictions. Although some projects are currently exploring the addition of wireless technology to restricted areas, it is not anticipated to be fully approved and operational anytime soon. As mentioned in the equipment section, the mobile platforms are typically manufactured with wireless capability, but the hardware was removed by the manufacturers at LANL's request. Tethered data cables are potentially an option, but they raise issues with management of the cable position during operation and increased dispersal of radioactive contamination that may be present on floors. Initial conversations with LANL facility personnel have established the strong opposition to tethers from their perspectives. The lack of wireless or tethers limits the use of real-time surveying and control; instead, the robot must be told in advance what tasks it will perform and must complete those tasks before operators can access its data. It is this limitation that necessitates the level of autonomy seen in this work. In this work, semiautonomy means the robotic system is told where to go, but the path it takes is determined by the system itself. Eventually advance directive could be programmed to be something as general as "perform this scan every night." The implications for robot operation are simply that the robot needs to be programmed to handle a variety of events and to gracefully handle unknown situations. Although the "upset" events in this work are simplified to avoid collisions, this is nevertheless one of the more significant things that could go wrong. Robotics that could be implemented at LANL need to prove their reliability and safety. Before robots are allowed to perform contact tasks, such as retrieving containers, they need to show that they can safely navigate the environment.

If a surveying robot were to eventually be approved to operate in the PF-4 vault, there are two other issues that would arise. First, the platforms in their current configuration in this work would not likely be able to open vault room doors. This would not largely

impact the project initially, as the system must be proven to operate safely and securely before being granted unescorted vault access. If a more fully automated system is eventually desired to directly retrieve and store material in vault rooms, the problems associated with having a robot open vault doors and shelving cages will have to be addressed.

Second, the way items are stored in the vault would provide challenges to scanning and handling them with automation. There are two metrics that limit the space within the vault. Storage locations in the vault are limited by the physical space available on which to place items. The shelving is divided into individual locations of varying sizes, and certain container sizes will only fit a subset of storage locations. Another limit is the total quantity of material. The presence of SNM requires controls in place to prevent criticality accidents. Each location has a physical footprint size that is combined with the criticality limit as a "crit footprint." The combined metric allows the colocation of several containers within one storage location. Based on current calculations that combine usage of physical space and usage of criticality space, the vault is over 80% full. Thus, there is a potentially significant number of locations that contain more than one item. How containers are placed into one location is not specified, and there may be containers occluding others. It is also possible for more than one source of radioactivity to be stored per container. Given these limitations, it would be nigh impossible to determine which item within one storage location would be responsible for a detected neutron. However, inventory systems track which items are in which locations, and any new information collected about stored material, even if the resolution is limited to generalized locations, is more information than is currently available.

3.2.2 Detector Aperture Equations

As laid out in the introduction, it would be useful to have some measure of directionality in a radiation detector that is used in storage locations at LANL. Given LANL's nature as an active research and production site, significant quantities of nuclear materials are stored, and they are often collocated. There are situations where it is possible to have one source interacting with a detector (see validation measurements with the detector in the next chapter), but this is unrealistic for any eventual robotic surveying system installed in the future. Selectively shielding a detector like a spyglass is a way to partially mimic what would be happening if there were only one source sending particles to the detector instead of many. This approximates what is already happening with detector physics, where only a fraction of the particles emitted by an isotropic source will enter the detector (Figure 47).

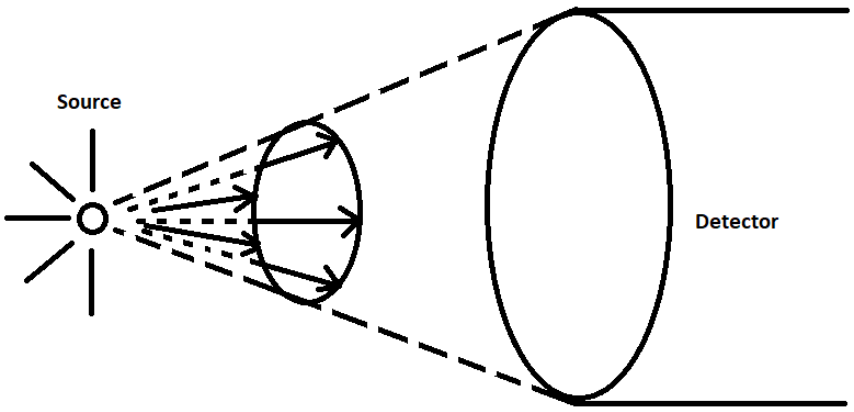


Figure 47. Solid angle subtended by detector from the source.

The solid angle, Ω, is defined as the number of particles per second emitted by space defined by contours of the source and detector aperture divided by the total number of particles per second emitted by the source. The solid angle can be calculated explicitly for some configurations of source and detector geometries (Tsoulfanidis & Landsberger, 2011). For an isotropic plane source with area A_S located a distance d away from a detector with aperture area A_d (Figure 48), the solid angle is:

$$\Omega = \frac{\int_{A_s} \int_{A_d} \left(\frac{S_0 dA_s}{4\pi r^2}\right) dA_d (\hat{\boldsymbol{n}} \cdot \boldsymbol{r}/r)}{S_0 A_s}$$

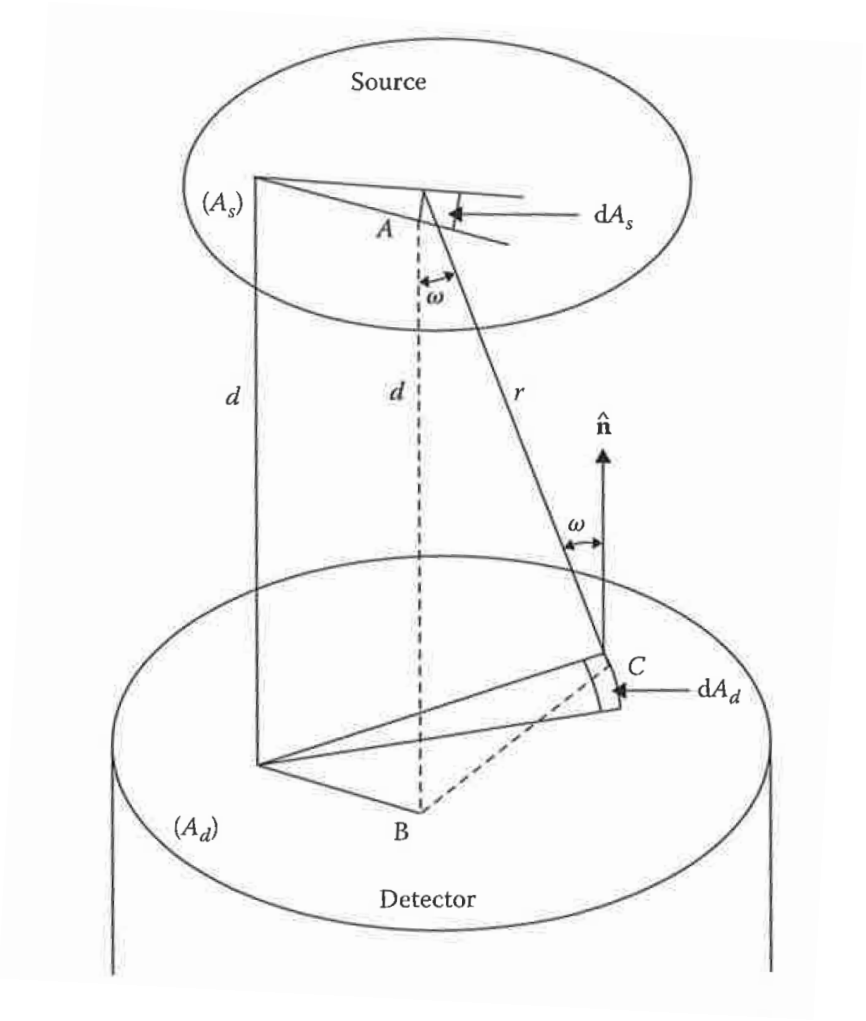


Figure 48. Parallel plane source and detector geometry (Tsoulfanidis & Landsberger, 2011).

However, as $(\hat{\boldsymbol{n}} \cdot \boldsymbol{r}/r)$ = cos ω, the solid angle equation can be simplified to:

$$\Omega = \frac{1}{4\pi A_{source}} \int_{A_{source}} dA_{source} \int_{A_{detector}} dA_{detector} \frac{\cos \omega}{r^2}$$

This geometry can be further simplified into a point source with a circular aperture (Figure 49).

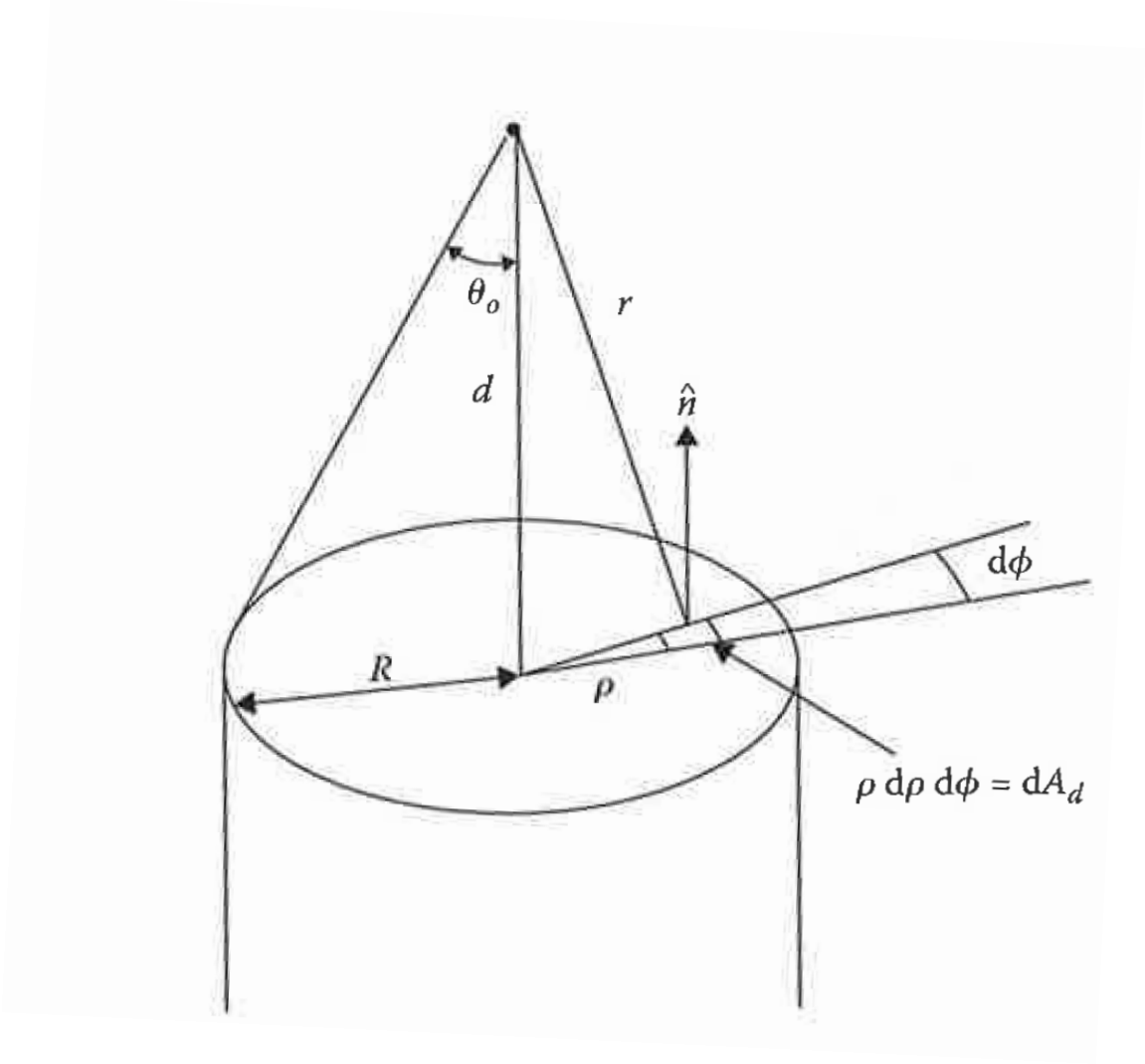


Figure 49. Point source with circular aperture detector geometry (Tsoulfanidis & Landsberger, 2011).

Reducing the previous definition for cos ω = *d*/*r*, the solid angle equation becomes:

$$\Omega = \frac{1}{2}\left(1 - \frac{d}{\sqrt{d^2 + R^2}}\right)$$

Further simplifications can be made from the geometry in Figure 49 for the value of θ_0:

$$\cos\theta_0 = \frac{d}{\sqrt{d^2 + R^2}}$$

Substituting this value into the previous solid angle equation, we now obtain:

$$\Omega = \frac{1}{2}(1 - \cos\theta_0)$$

When the source to detector distance is large relative to the aperture size (*R* << *d*), the value for $\cos\theta_0$ can be series expanded:

$$\frac{d}{\sqrt{d^2 + R^2}} \approx 1 - \frac{R^2}{2d^2} + \frac{3R^4}{8d^4} - \frac{5R^6}{16d^6} + \cdots$$

Substituting the first two terms of this series, the solid angle equation becomes:

$$\Omega = \frac{1}{2}\left(1 - \left(1 - \frac{R^2}{2d^2}\right)\right) = \frac{R^2}{4d^2} = \frac{\pi R^2}{4\pi d^2}$$

$$\Omega = \frac{detector\ aperture}{4\pi d^2}$$

Although it may confirm what already seems intuitively true, this result is important because it confirms the strong relationship between solid angle and detector aperture. A neutron detector spyglass is the creation of an artificial aperture on the detector. The field of view is directly related to the area of interest when observing a source. Calculating the solid angle for more complicated geometries becomes much more complex and usually must be solved via numerical integration or approximations. This complexity is where simulation such as with MCNP is very useful. MCNP can help examine how neutrons are entering the aperture versus how they are being shielded from other directions.

3.2.3 Neutron Sources

To discuss shielding the neutrons, it is first useful to discuss how these neutrons are generated. Given the presence of SNM at LANL, many of the isotopes stored can undergo spontaneous fission. However, it is a common misperception that neutrons generated from spontaneous fission are the only neutrons of concern. The primary decay mode for most plutonium isotopes is alpha particle emission. These alpha particles produced by radioactive decay can then go on to (α, n) reactions. The probability of alpha decay, spontaneous fission, half-life, mass defect, and molecular weight for certain isotopes of plutonium, americium, and uranium are shown in Table 3.

Table 3. Alpha decay and spontaneous fission rates for selected isotopes (Davis, Kornreich, & Lambert, 2011).

Isotope[7]	% α	% SF	τ (y)	Δm (MeV/c^2)	MW (g/mol)
Pu-238	100	1.9×10^{-7}	87.7	46.1647	238.049560
Pu-239	100	3.1×10^{-10}	24,110	48.5899	239.052163
Pu-240	100	5.7×10^{-6}	6561	50.1270	240.053814
Pu-241	0.00245	2.4×10^{-14}	14.29	52.956	241.056851
Pu-242	100	5.5×10^{-4}	3.75×10^{5}	54.7184	242.058743
Am-241	100	3.6×10^{-10}	432.6	52.936	241.056829
U-232	100	8.5×10^{-20}	68.9	34.61	232.037155
U-233	100	6.0×10^{-9}	1.592×10^{5}	36.92	233.039635
U-235	100	7.0×10^{-9}	7.04×10^{8}	40.9205	235.043930
U-238	100	5.45×10^{-5}	4.468×10^{9}	47.3089	238.050788

Various elements have higher probabilities of interacting with alpha particles to produce neutrons, and the yield is energy dependent (Table 4).

Table 4. Yields from (α, n) reactions (modified from Reilly et al., 1991).

Element (Natural Isotopic Composition)	Neutron Yield per 10^6 Alphas of Energy 4.7 MeV (^{234}U)	Neutron Yield per 10^6 Alphas of Energy 5.2 MeV (av. Pu)
Li	0.16 ± 0.04	1.13 ± 0.25
Be	44 ± 4	65 ± 5
B	12.4 ± 0.6	17.5 ± 0.4
C	0.051 ± 0.002	0.078 ± 0.004
O	0.040 ± 0.001	0.059 ± 0.002
F	3.1 ± 0.3	5.9 ± 0.6
Na	0.5 ± 0.5	1.1 ± 0.5
Mg	0.42 ± 0.03	0.89 ± 0.02
Al	0.13 ± 0.01	0.41 ± 0.01
Si	0.028 ± 0.002	0.076 ± 0.003
Cl	0.01 ± 0.01	0.07 ± 0.04

As Table 4 notes, the average decay alpha energy is dependent upon isotope; the average alpha energy for uranium-234 is 4.7 MeV, and overall for plutonium it is 5.2 MeV. The average energy of the neutron produced from the reaction depends on the target element (Table 5).

Table 5. Average (α, n) neutron energy per target by source isotope (modified from Davis, Kornreich, & Lambert, 2011).

Target	Pu-238	Pu-239	Pu-240	Pu-241	Pu-242	Am-241	U-232	U-233	U-235	U-238
Be	4.994	4.830	4.833	4.758	4.757	4.990	4.898	4.743	4.716	4.726
F	1.300	1.233	1.234	1.172	1.171	1.298	1.264	1.151	0.9541	0.8328
C	4.707	4.697	4.708	4.456	4.450	4.713	4.737	4.423	4.265	4.216
O	2.373	2.265	2.269	2.199	2.202	2.376	2.319	2.181	2.083	1.993
Al	1.091	1.020	1.022	0.9730	0.9724	1.090	1.059	0.9540	0.7739	0.6831
Cl	0.6909	0.5451	0.5503	0.3714	0.3680	0.6882	0.6249	0.3165	0.1073	--

Although there are variations in the average energy of neutrons produced from spontaneous fission and (α, n) reactions, there is some overlap, as shown in Figure 50 (for

plutonium-238, plutonium-240, and curium-244, but this holds true for other similar isotopes).

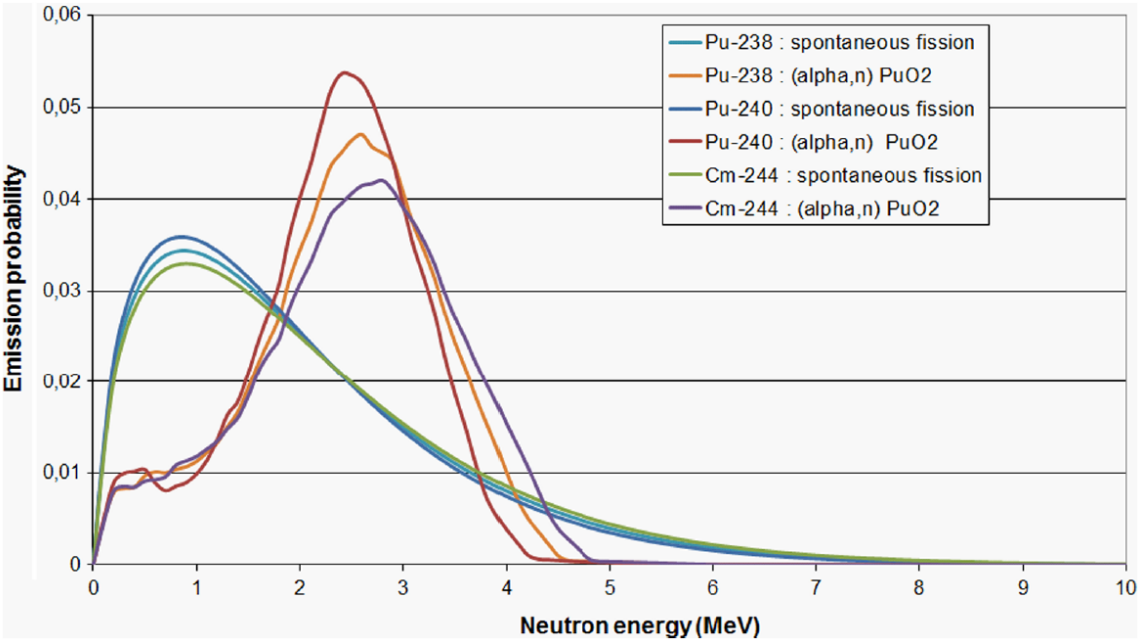


Figure 50. Neutron energy spectra on plutonium and curium isotopes (Pérot et al., 2018).

Californium-252 deserves special attention because it has a similar energy spectrum to neutrons produced from plutonium interactions (Figure 51); it can be more readily available than plutonium; it produces a higher number of neutrons per gram relative to other isotopes (and produces more neutrons per fission than others at 3.73 neutrons on average [Shultis & Faw, 2010]); and it has a higher spontaneous fission rate (its fission branching ratio is 3.09% with a half-life of 2.645 years [KAERI, n.d.]). These qualities enable Cf-252 to be used as a substitute when performing experiments and simulations; it was the source used in this work for MCNP and validation of the manufactured neutron shield.

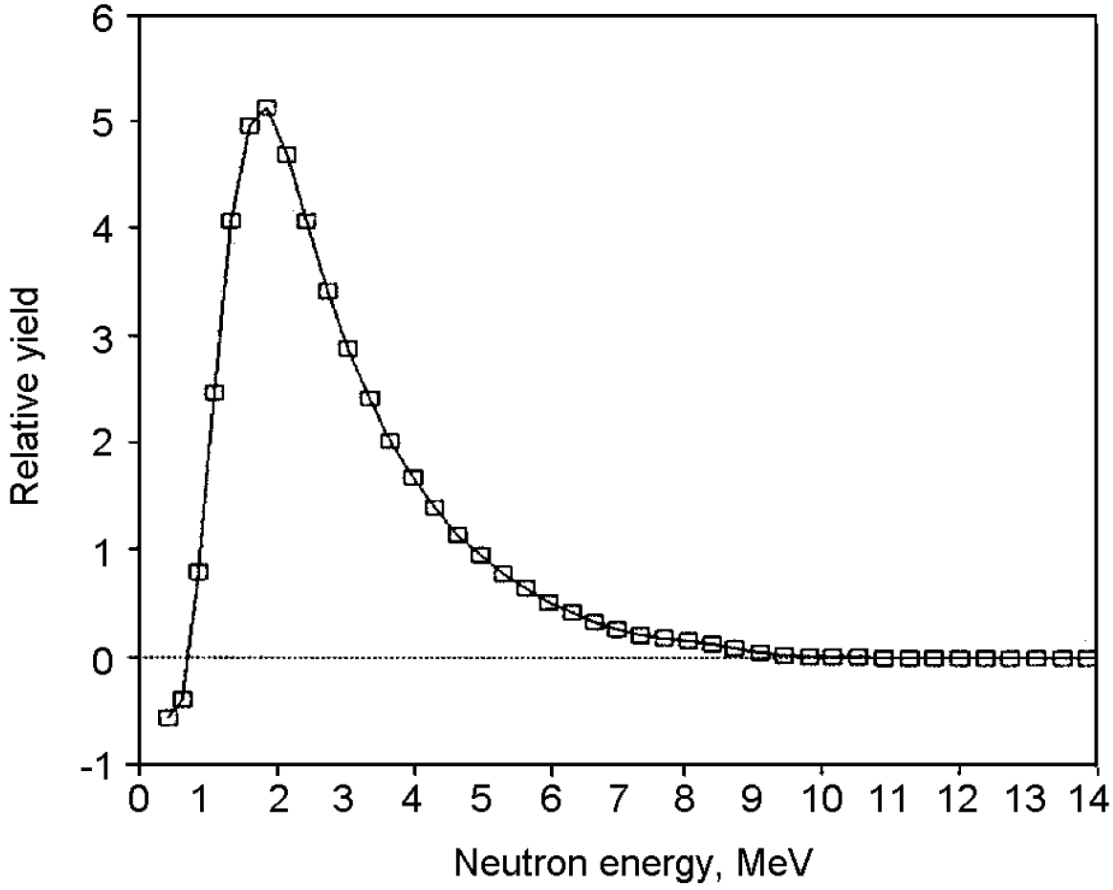


Figure 51. Neutron energy spectrum of Cf-252 spontaneous fission (Park, 2003).

3.2.4 Neutron Shielding

Because neutrons are neutrally charged, they interact with nuclei through nuclear forces and do not have to penetrate Coulomb barriers generated by both the negatively charged electron cloud and the positively charged nucleus. The two main types of neutron

interactions are scattering and absorption. Scattering may be elastic or inelastic, and the interaction is represented as (n, n) because both the nucleus and neutron are intact after the collision. The total kinetic energy is conserved in elastic scattering, and part of the incident neutron energy is transferred to the nucleus, which puts it in an excited state, in inelastic scattering (this energy later being released as gamma rays as the nucleus returns to its ground state). Scattering is responsible for the slowing down or "moderation" of neutrons. Neutrons born from fission reactions have an average energy around 1 to 2 MeV. Cross sections for fission and absorption generally increase as neutron energy decreases. The increased probability of interaction for slower neutrons ("thermal" being defined at 0.0253 eV) is the reason for moderating material used in nuclear power reactors and shielding applications. Moderating material is usually water, graphite, or other hydrogenous materials.

Neutron absorption reactions cover many different types of interactions, such as (n, p), which produces a proton (n, α), which produces an alpha particle (n, 2n), which creates two neutrons from one (n, γ), which produces a gamma ray or a fission reaction in which a neutron is absorbed into a nucleus that splits into daughter fragments and a variable number of additional neutrons (Tsoulfanidis & Landsberger, 2011). To create a neutron shield around a detector, the best approach is to slow down neutrons so they can then be absorbed.

3.2.5 Shielding Material

A few materials can be considered for neutron detector shield construction. With a generalized 1/v trend for neutron interaction cross sections, slowing down neutrons makes them more likely to interact. The shield should be able to moderate the neutrons but then absorb them to prevent as many as possible from reaching the detector along unwanted

directions. A well-known neutron moderator is polyethylene, which is used because of its hydrogenous content (Figure 52). Polyethylene also has the advantage of being inexpensive and easy to machine into custom shapes.

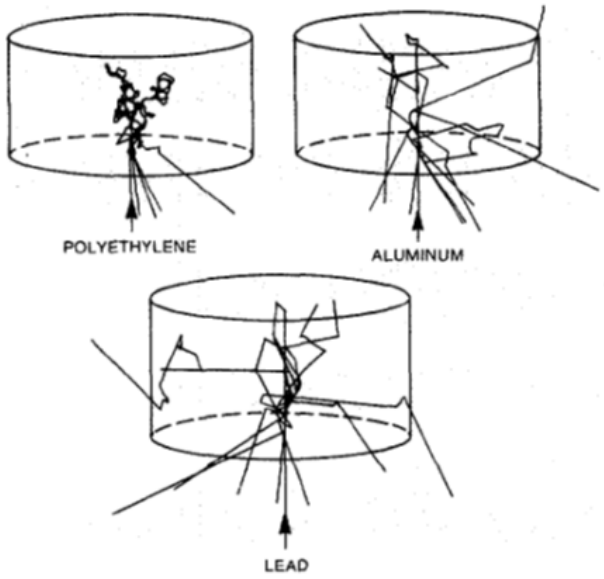


Figure 52. Monte Carlo simulations of 1 MeV neutrons entering cylinders of polyethylene, aluminum, and lead (Rinard, P., n.d.).

Cadmium, boron, and lithium are three commonly used neutron absorbers. Cadmium is excellent at absorption but is potentially toxic and can produce significant capture gammas. Boron and lithium also produce capture gammas, but both are available already incorporated into commercial polyethylene products. Lithium's capture gamma is weaker than that of boron's, but a lithium shield would need to be thicker due to its overall lower capture cross section compared to boron. The total cross sections for the two most abundant isotopes of cadmium, boron, and lithium are shown in Figure 53.

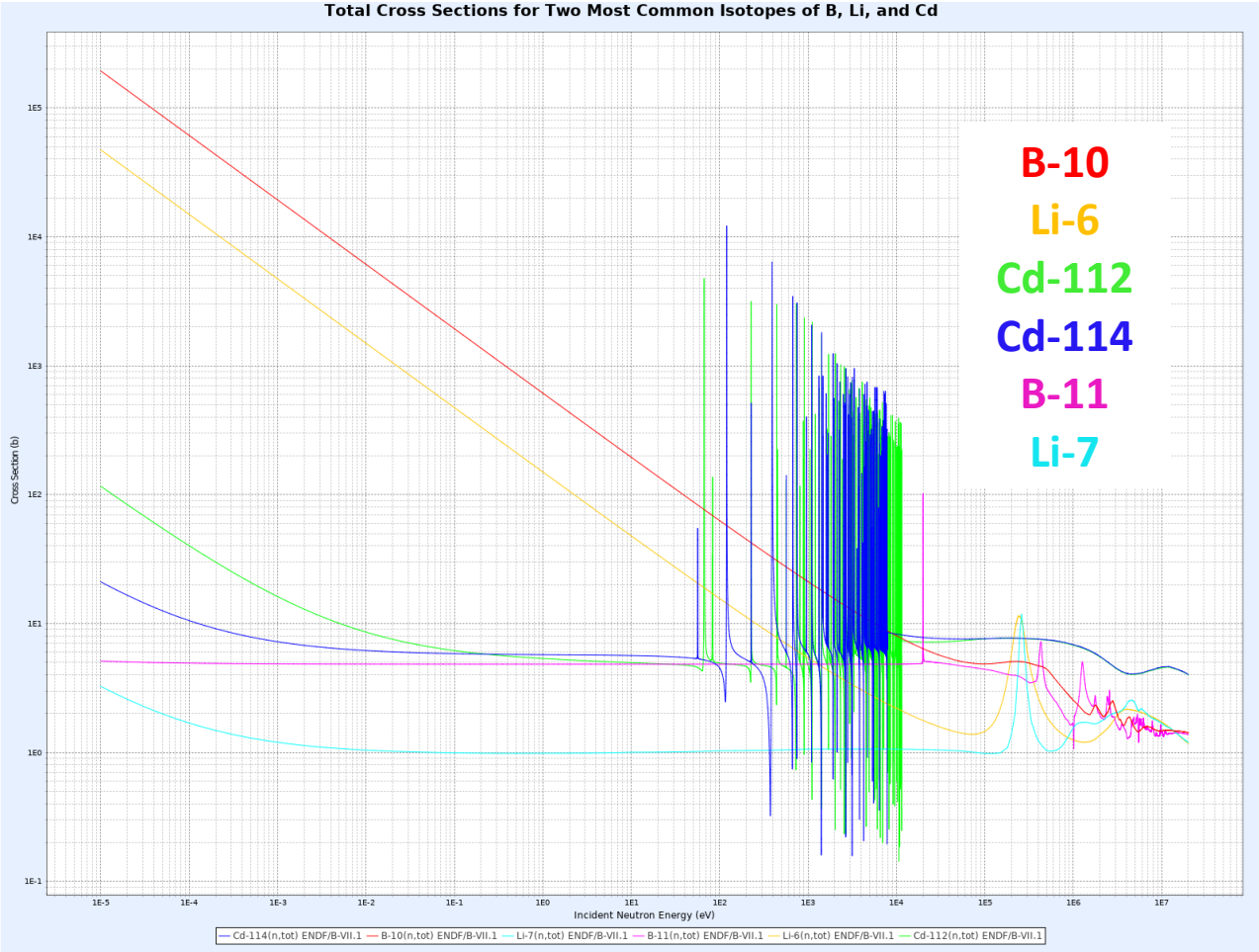


Figure 53. Total cross sections for two most abundant isotopes of boron, lithium, and cadmium (KAERI, n.d.).

The mobile platforms discussed in this chapter have relatively large payload capability, but neutron energies encountered in this application can range up to several MeV. To prevent the shielding thickness from getting too unwieldy, it was anticipated that borated polyethylene would make a good initial shielding material for this application. The simulation of the shielding is contained within the following chapter.

3.2.6 Counting Statistics

Radioactive decay is inherently a random process. Half-lives can be defined, but when taking measurements, there is always statistical fluctuation. Because decay is a binary process (success or failure), and the decay of one nucleus is independent from that of another nuclei, the distribution of results from the decay of a group of nuclei can be represented by a binomial probability distribution:

$$P(x) = \frac{n!}{(n-x)!\,x!} p^x (1-p)^{n-x}$$

Where *n* is the number of trials, each of which has a probability of success *p*, and P(*x)* is the probability of counting *x* successes. This is analogous to tossing coins and noting if they land heads or tails. Because the distribution is normalized, the mean value of the distribution is given by:

$$\bar{x} = \sum_{x=0}^{n} xP(x) = pn$$

In nuclear counting experiments where the number of nuclei is large in comparison to their half-life, the results can be characterized according to a Poisson distribution. The Poisson distribution is itself a reduction of the binomial distribution when the probability of "success" (or decay) is small but constant:

$$P(x) = \frac{(\bar{x})^x e^{-\bar{x}}}{x!}$$

When mean value of the distribution is large, in addition to the probability of success being small, the Poisson distribution further simplifies to the Gaussian distribution:

$$P(x) = \frac{1}{\sqrt{2\pi\bar{x}}}\exp(-\frac{(x-\bar{x})^2}{2\bar{x}})$$

The predicted variance of this distribution is given by:

$$\sigma^2 = \sum_{x=0}^{n}(x-\bar{x})^2 P(x)$$

For the Poisson and Gaussian distributions the predicted variance simplifies to:

$$\sigma^2 = pn = \bar{x}$$

Thus, the standard deviation is the square root of the predicted variance and the square root of the mean value:

$$\sigma = \sqrt{\bar{x}}$$

This result is important because when measuring radiation, one must be certain that the data being collected is of statistical significance (further clarification of the equations above can be found in [Knoll, 2000]). The minimum detectable activity (MDA) for a counting system can be calculated according to the Currie method (1968). Currie defined limits useful when analyzing the statistics of a signal: Critical Level, L_C, is the signal level

above which a recorded signal may be called "detected," and Detection Limit, L_D, is the *a priori* expected "true" signal taking statistical variance into account. It is desirable to set these limits in such a fashion that both false positives and false negatives are minimized (Figure 54).

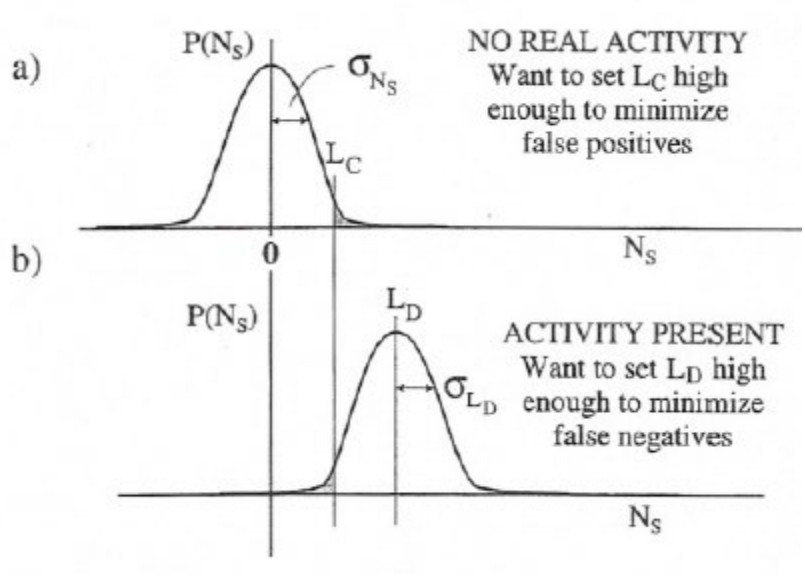


Figure 54. Distributions expected for the net counts N_S for the cases of A) no activity present and B) real activity is present. L_C is the critical level or "trigger point" of the counting system, and L_D corresponds to the mean number of net counts needed for MDA (modified from Knoll, 2000).

In order to have a confidence level of 95% (that is, false positive probability is no larger than 5%), Currie (1968) defines the following formula:

$$MDA = L_D = k^2 + L_C = 2.706 + 4.653\sigma_{N_B}$$

Where *k* defines the abscissas of the standardized normal distribution for the desired confidence level and σ_{N_B} is the standard deviation of the background count signal. A

background count was performed on the bare Bridgeport Instruments neutron detector in a laboratory setting. The count was taken over 30 minutes and the background total count was 10,447 (rate 5.804 ± 0.057 counts/sec). The MDA (95% confidence level) in this scenario is as follows:

$$MDA = 2.706 + 4.654\left(\sqrt{10447}\right) = 478.29\ counts$$

Many other applications for neutron detectors are often concerned with detecting small numbers of the particles (see Chapter 2). The counting statistics in this work benefit from the relatively large population of neutrons when examining stored nuclear materials. The issue, as will be seen in the following chapter, is not a dearth of neutrons to detect but, rather, being able to block enough of them to gain some small degree of directionality to the neutron counts.

Chapter 4: Findings

This chapter describes the work that was performed to set up, model, characterize, and test the equipment and shield for the detector. Security approvals may seem a small matter, but they require a large amount of work in a sensitive environment like LANL. Much of the initial work involved obtaining these approvals for the robotic platforms and demonstrating their safe operation; then the detector modeling and testing followed.

4.1 Pioneer LX Approvals

When the Pioneer LX mobile platform first arrived at LANL, it underwent an initial security inspection, as is customary with items entering the Laboratory. It was then delivered to the uncleared side of the RLUOB building at TA-55. It took one month to obtain approval for the Activity Security Plan (ASP) allowing operation within a predefined area. The system software was used to map the area (a small room and hallway containing cubicle offices) and waypoints selected in the area to demonstrate the autonomous navigation capabilities of the platform. Demonstrations were performed for various groups to showcase the technology and safety features. A popular example was testing of the collision avoidance by jumping in front of the platform as it was navigating between waypoints, thus showing that the platform would not run into the demonstrator and could calculate another path around the obstacle to the intended waypoint.

In an effort to move toward approval to operate the robot within PF-4, the LANL Information Security Site Manager (ISSM) and the National Nuclear Security Administration (NNSA) Los Alamos Field Office (LAFO) Authorizing Official evaluated the Adept Pioneer LX from an information security perspective. They approved of the platform not utilizing any wireless networking (such as Wi-Fi or Bluetooth) and that it was using lasers instead of visual camera feedback to orient itself and navigate. Although it

would maneuver in an area that had classified computing (PF-4), it would not connect to any classified computer systems nor store any classified data (a distinction that will not necessarily hold true in the future if the platform is used in the vault). To remove data from the platform, it must connect via USB or local Ethernet. In many senses, a mobile platform that is not wirelessly communicating is like a computer peripheral. This placed the Pioneer into a similar class as scientific equipment (detectors, oscilloscopes, spectrometry instrumentation, and the like), which needs no specific information security documentation. Instead, the ISSM requested an operational checklist (an ASP) that would clearly demonstrate how the device would be used, its area of operation, and what it could and could not connect to. If additional networking capabilities were to be added in the future, additional information security testing, documentation, and approvals would be required. After approvals were obtained for this new ASP, the Pioneer was granted approval to be relocated into the basement of PF-4. The room that the Pioneer is now located in (previously shown in Figure 44) houses some process control equipment and apparatus for container surveillance testing. No radiological sources are used in this room. However, this room does have shelving with empty containers (staging for container surveillance, Figure 55). As before, the room was mapped, and waypoints were set to demonstrate that the robot could autonomously navigate the room, avoid previously unknown obstacles, and maneuver to the shelving to demonstrate what an inventory inspection behavior could look like. Throughout testing, no collisions were observed or could be induced.

Figure 55. Container shelving in Pioneer LX room.

4.2 VECTOR APPROVALS

Similar to the Pioneer LX approval process, the Vector platform also underwent initial inspection when it first arrived at the Laboratory. Many other reviews and procedures had to be put in place to allow operation at TA-35. In order to provide greater longevity to the overall system, an additional battery was installed on the platform to provide power to the arm and its control box. Review and approval had to be obtained from LANL Electrical Safety Officers (ESOs) for the base platform, the arm, the arm control box, and the additional battery. Due to the capacity of the battery (100 amp-hours dry-cell absorbed glass mat lead acid), only ESOs were allowed to handle and connect it to the system. Additionally, an Integrated Work Document was created, approved, and implemented to govern the safe operation of the system and its equipment. Such a document is required by the Laboratory to perform any work in a hazardous environment. Hazards from the system are moving parts (collision, pinch points, and the like) and electrical components; additionally, the area of operation can have radiological sources.

Custom parts were machined at LANL to integrate all the peripherals. An aluminum base mounting plate and hollow cylindrical column were fabricated to provide an attachment point for the Universal Robots arm (Figure 56). Other components were then attached to the base mounting plate. To attach the neutron detector and shield, two brackets were machined to attach to the front of the mounting plate (Figure 57 and Figure 58).

Figure 56. Machined aluminum base and arm mounting plate.

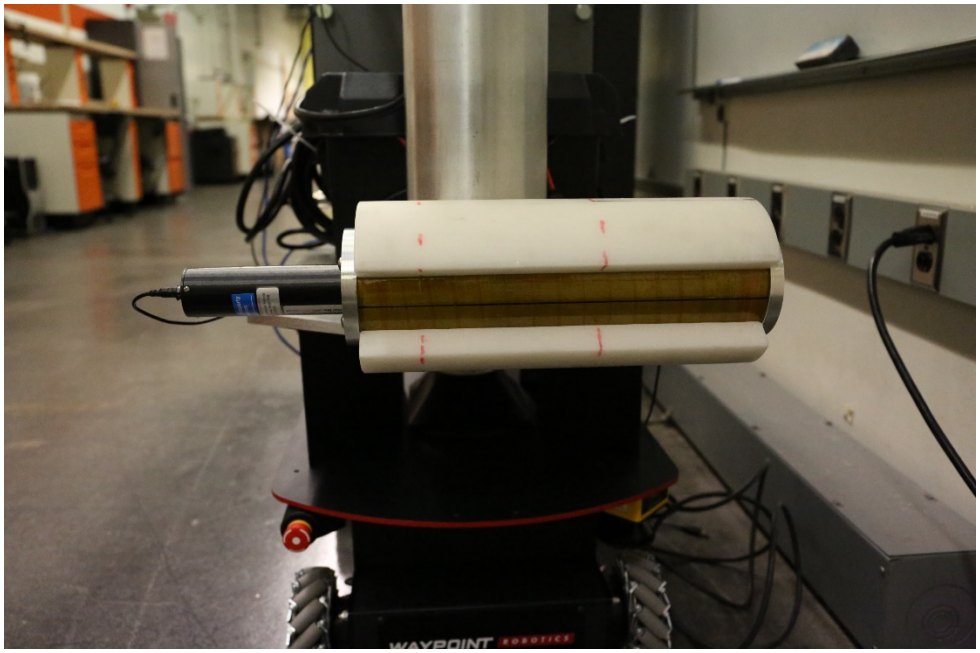


Figure 57. Detector and shield assembly mounted on front of base plate.

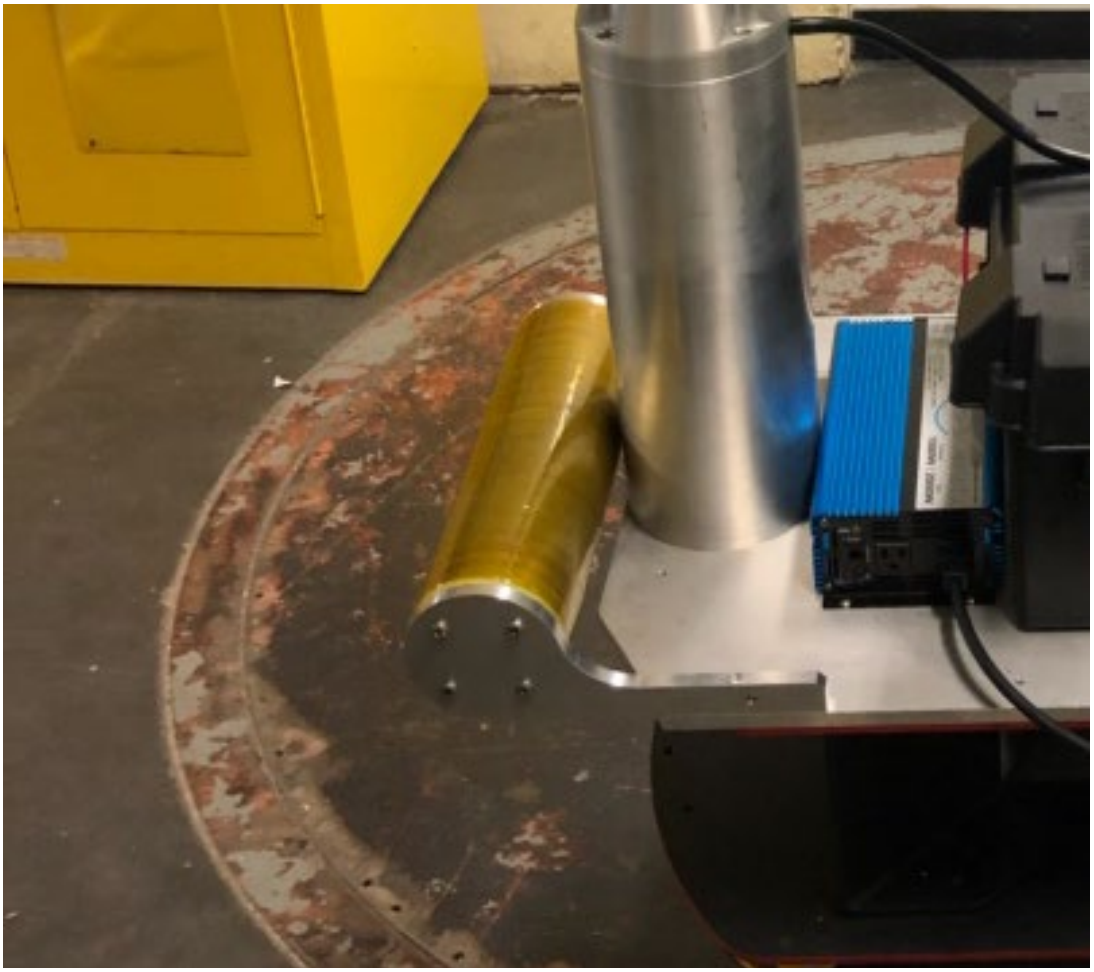


Figure 58. Partially constructed shield mounted on base plate showing the side detail of shield mounting bracket.

4.3 Detector Characterization

The same model of Bridgeport Instruments neutron detector was also procured at the Nuclear Engineering Teaching Lab at the University of Texas at Austin (UT). Although the body style was slightly altered compared to the detector at LANL, the internal mechanics are the same. As shown previously in Figure 29, the detector is a long cylinder

that contains the operating electronics on one end and the active volume of the detector at the other. The UT detector is 53.34 cm long while the LANL detector is 51.43 cm long; the variation is in the outer tubing and electrical connectors (the UT detector has an 8-pin EN3 connector in addition to the USB connector). The detecting elements of the detectors are roughly 30 cm long. It logically follows that the area of highest efficiency of detection would be located along the active volume (versus over the electronics side), but several experiments were performed to quantify this performance as well as to characterize the other spatial and rotational dependencies of the detector. These tests were performed using a PuBe neutron source and counted with the eMorpho software that is included with the Bridgeport Instruments detector.

4.3.1 Linear Dependence

This test was to assess the linear dependence of the neutron detector. The source was moved along the length of the detector and counts were measured. Seven sets of counts were taken at each point for 2.5 minutes each, and the counts were averaged. The distance is defined in terms of the offset from the electronics end of the detector (0 cm, Figure 59) and position of the source incremented by 2 cm. As expected, the detector exhibits a maximum efficiency over roughly the center of the active volume (Figure 60). Deadtime was less than 3% during this test and count rate error was ±1.8 counts per second (cps).

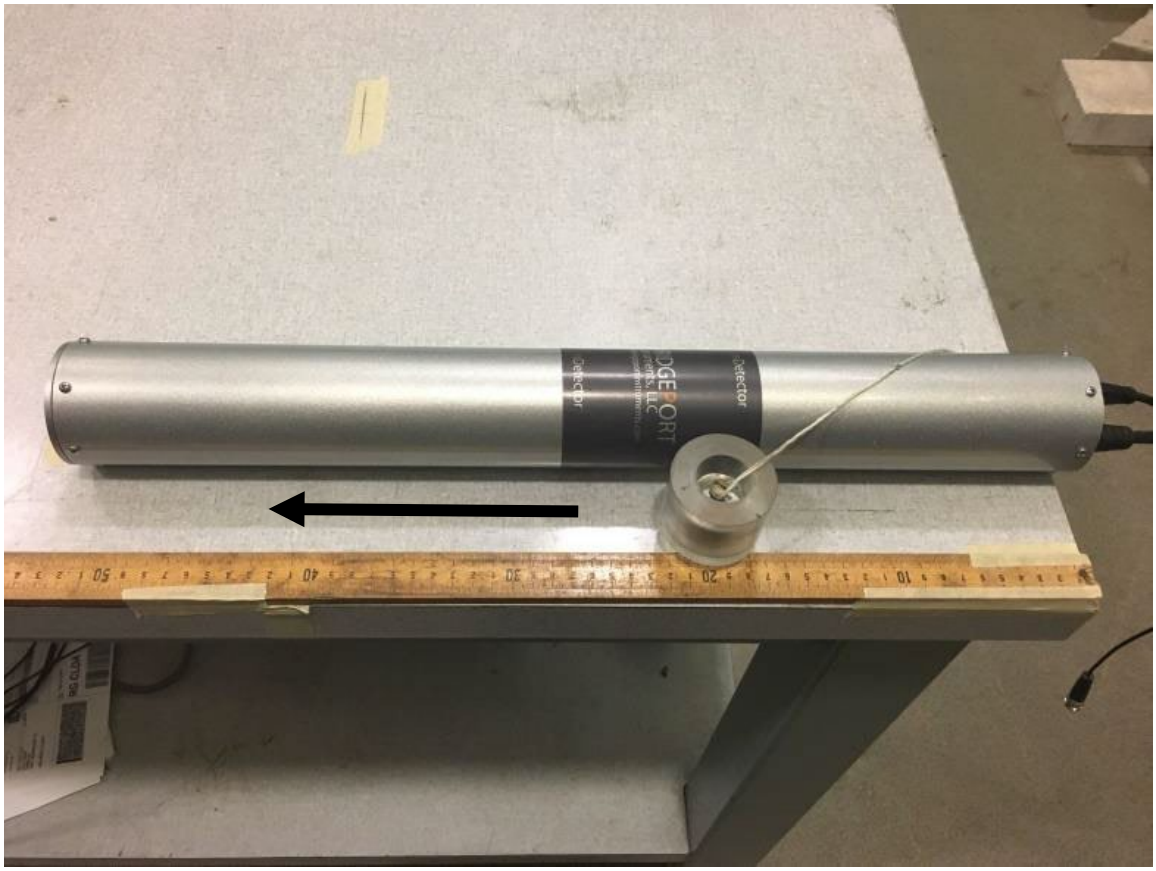


Figure 59. Linear dependence experimental setup.

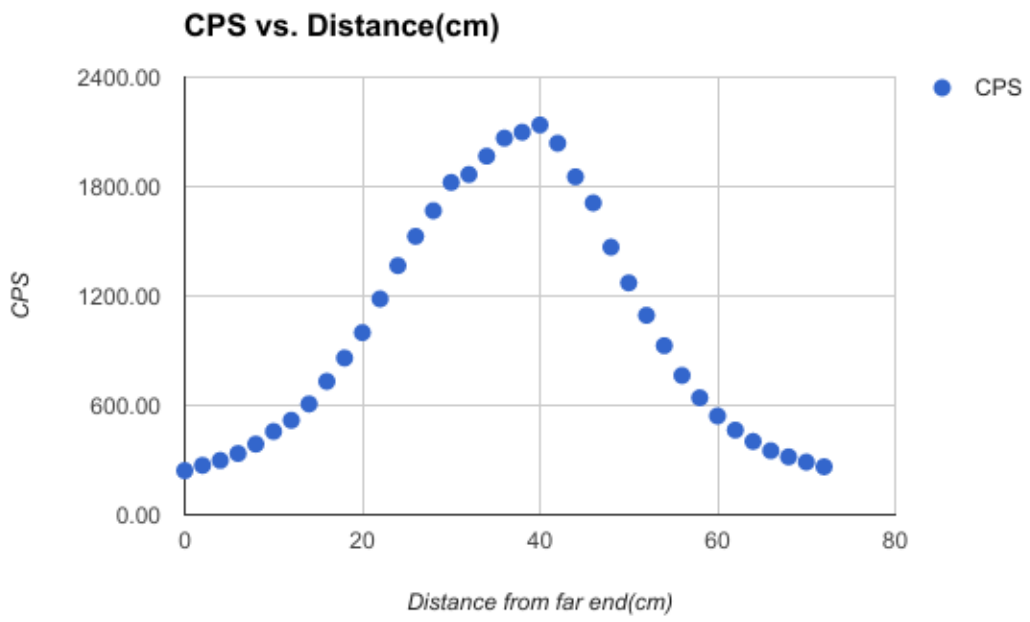


Figure 60. Count rate results from linear dependence test.

4.3.2 Long Axis Rotation Dependence

This test was to assess the rotational dependence of the neutron detector along the long axis. This experiment was ran in two configurations with the source placed 100 cm in front of the detector along the long axis and with the source 100 cm from the long axis positioned orthogonally (Figure 61). Markings were placed on the surface of the detector to control for position to adjust rotation in 20 degree increments (Figure 62).

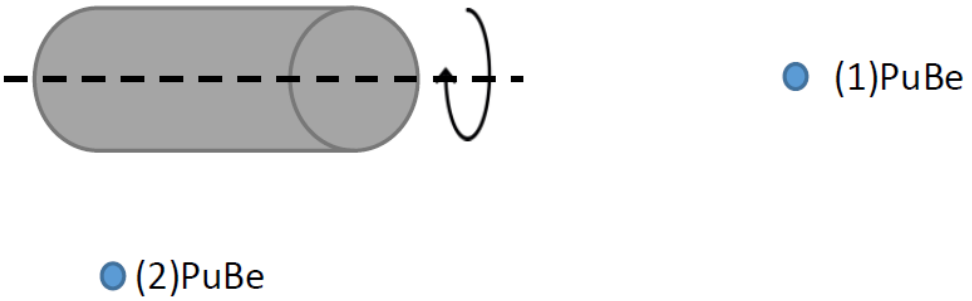


Figure 61. Detector rotation and source positions for long axis rotation dependence test.

Figure 62. Detector and source in position 1 for long axis rotation dependence test.

During the first position along the same axis as the long axis, the count rate remained relatively stable. The average count rate was 73.771 cps with a range of 2.9341 cps. Deadtime remained less than 0.09% during this test and error within ±1.7 cps (Figure 63).

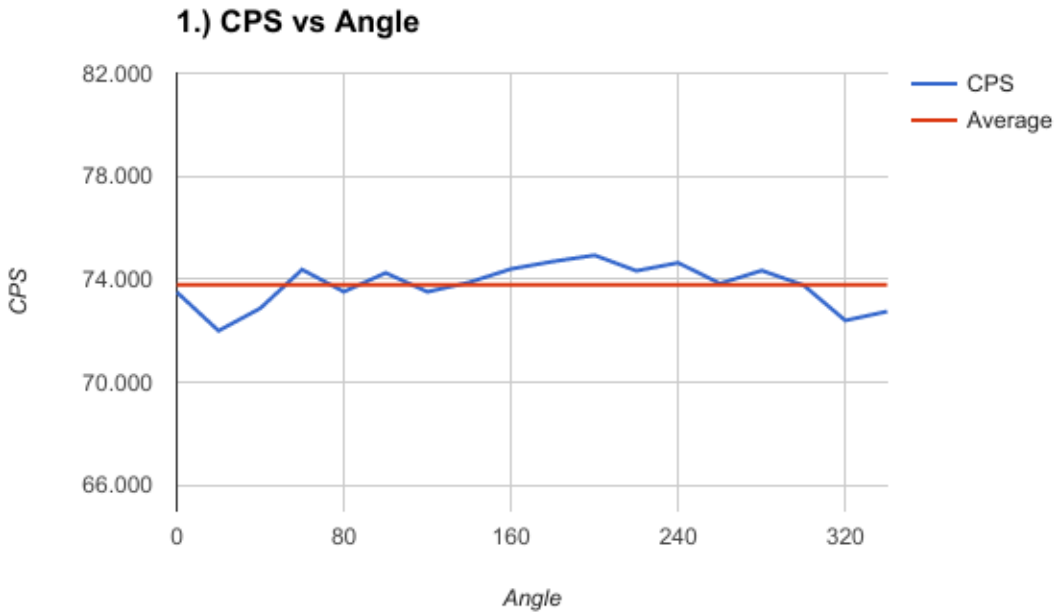


Figure 63. Count rate results for long axis rotation dependence test in position 1.

The count rate again remained relatively stable during the second position orthogonal to the long axis. The average count rate was 95.930 cps with a range of 3.403 cps. Deadtime remained less than 0.09% and error within ± 1.7 cps (Figure 64). These two tests show that there is little concern for the rotational position of the detector along its long axis when taking measurements.

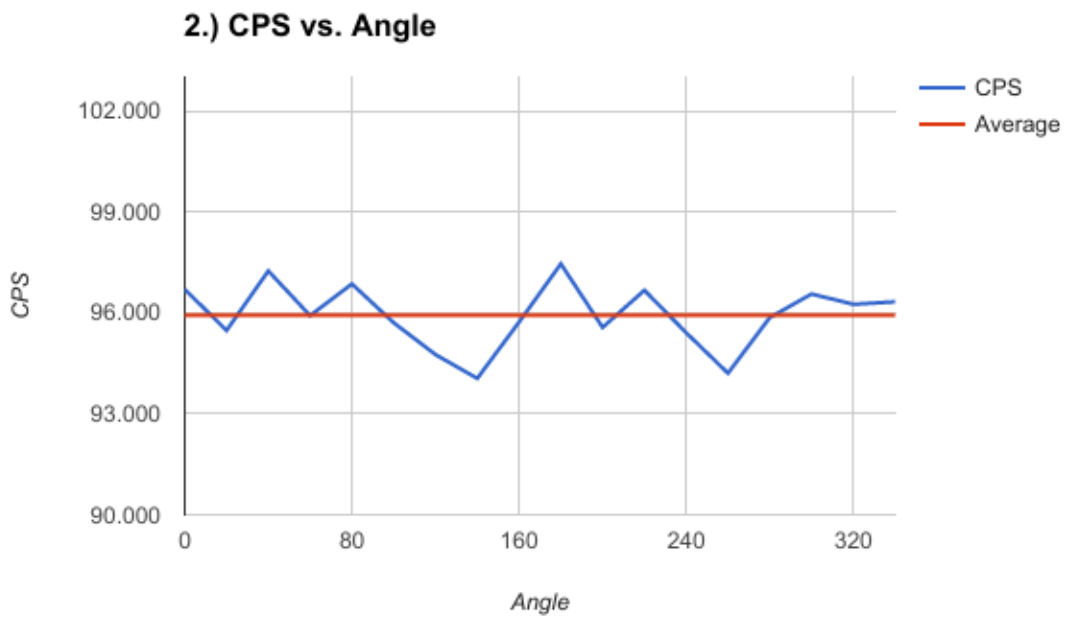


Figure 64. Count rate results for long axis rotation dependence test in position 2.

4.3.3 Short Axis Rotation Dependence

Similar to the long axis, measurements were taken with the source at three positions while the detector was rotated about the short axis in the Z/up direction (Figure 65). The angle was varied over 180 degrees in 5 degree increments. The position of 0 degrees was defined as the source placed a distance away from the detector along the detector's long axis (Figure 66). Because it was previously established that the detector shows a loss of efficiency when the electronics end of the detector is closer to a source, the detector was only rotated ±90 degrees from the 0 degrees position (Figure 67). The experiment was conducted with the source at 75 cm, 100 cm, and 140 cm.

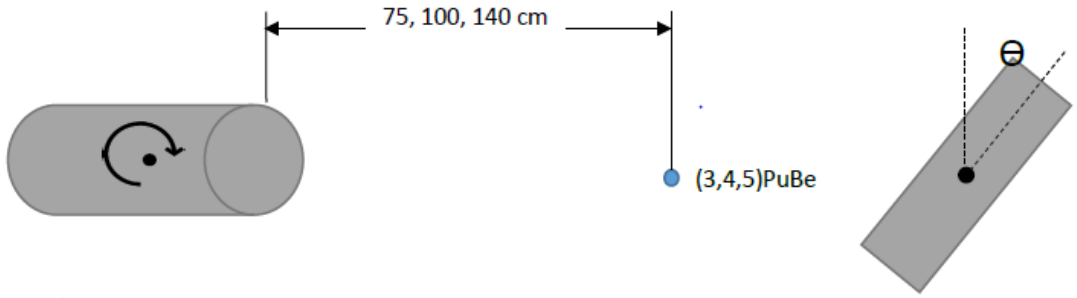


Figure 65. Detector rotation and source positions for short axis rotation dependence test.

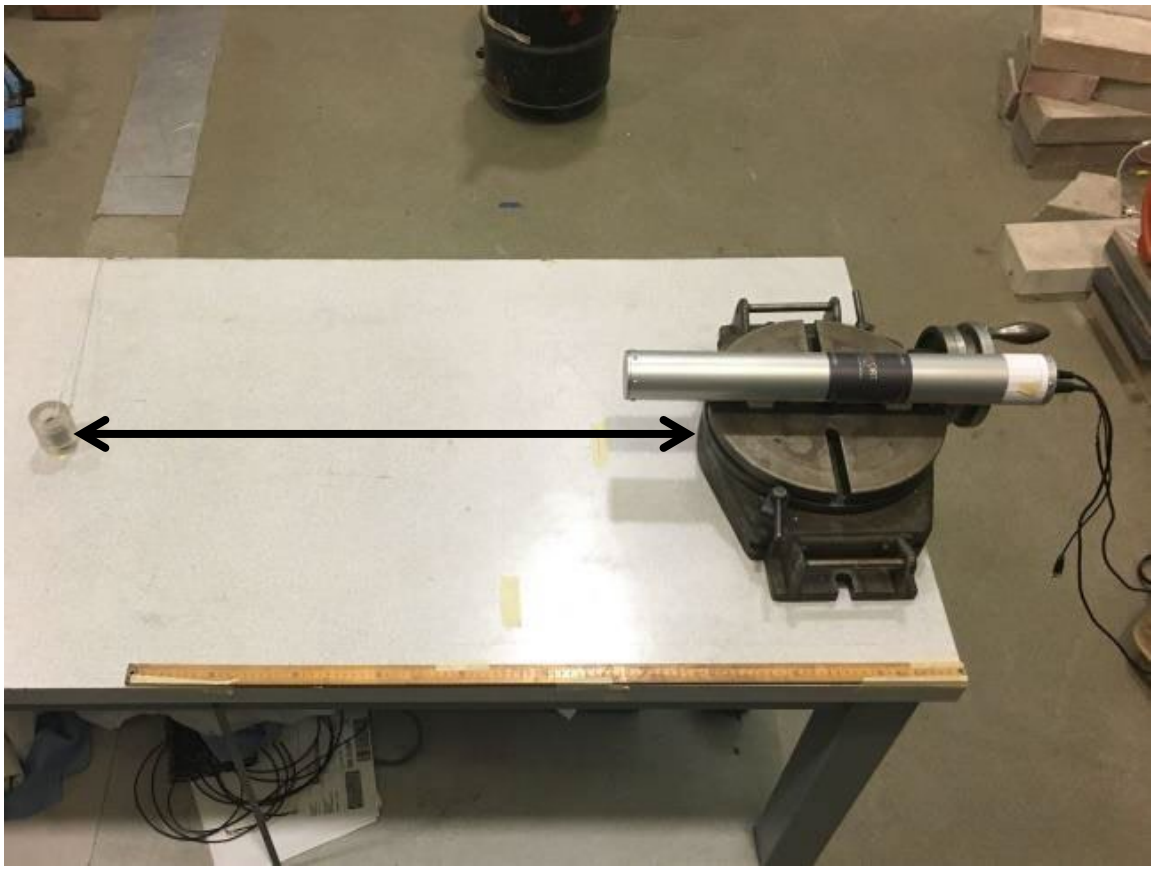


Figure 66. Detector at 0 degrees and source at position 5, 140 cm, for short axis rotation dependence test.

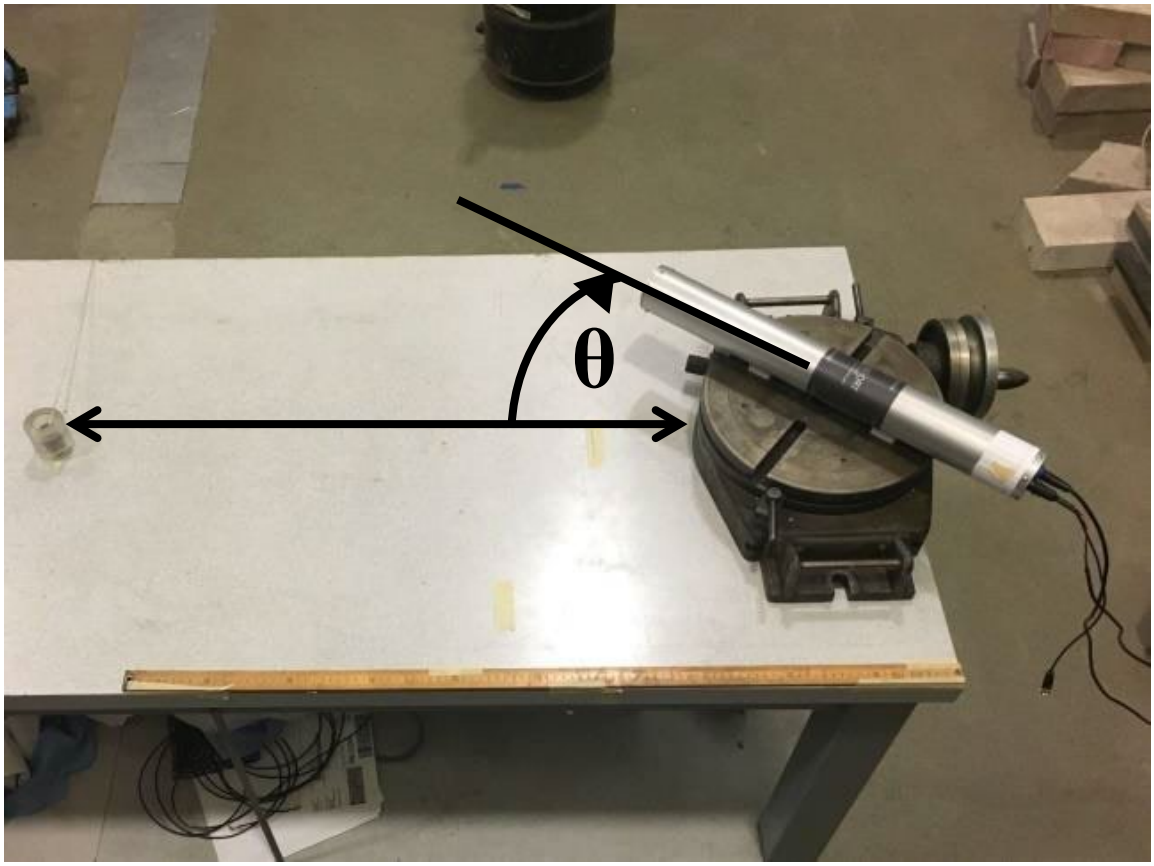


Figure 67. Detector at the 40 degrees to the right position and source at position 5, 140 cm, for short axis rotation dependence test.

At the third position, 75 cm, the deadtime was less than 0.12% and error was within ±1.8 cps (Figure 68). At the fourth position, 100 cm, the deadtime was less than 0.10% sand error was within ±1.7 cps (Figure 69). At the fifth position, 140 cm, the deadtime was less than 0.08% and error was within ±1.7 cps (Figure 70).

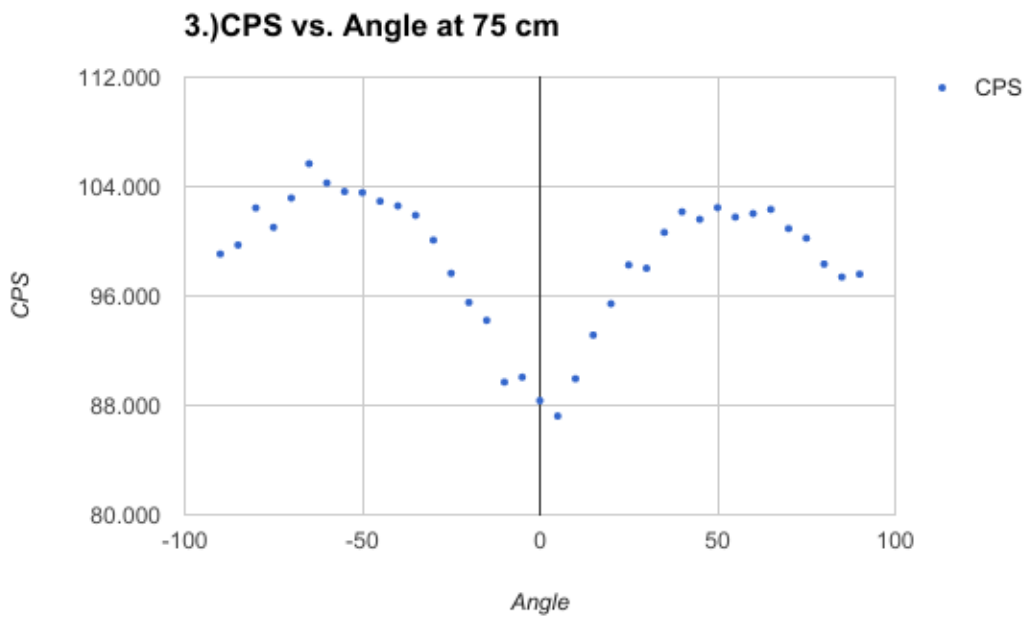


Figure 68. Count rate results for short axis rotation dependence test at position 3 (75 cm).

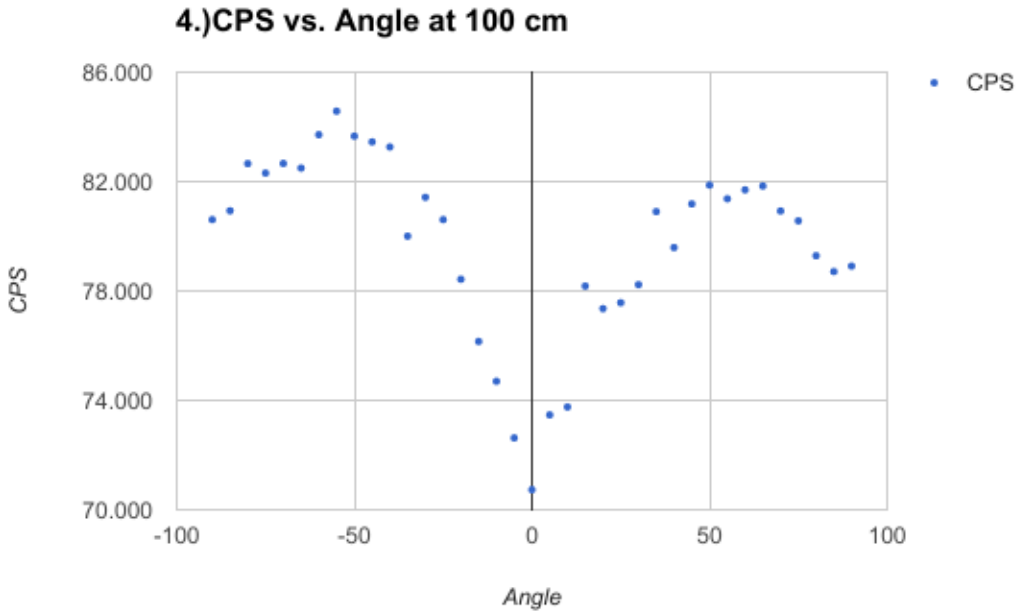


Figure 69. Count rate results for short axis rotation dependence test at position 4 (100 cm).

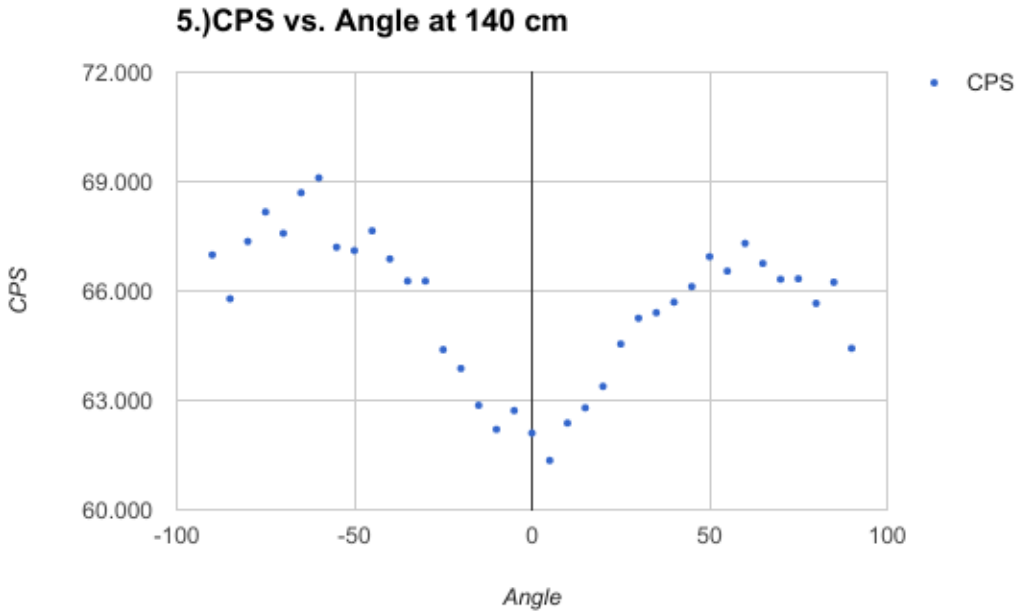


Figure 70. Count rate results for short axis rotation dependence test at position 5 (140 cm).

As the detector is rotating about its short axis, several factors are actually changing during the experiment: the solid angle to which the active volume is being exposed, the distance between each point in the active volume and the source, and the efficiency of detection. The results from the three positions were normalized and compared to each other to assess the effects of distance (Figure 71). It can be seen that as distance increases, the effects of the rotation on the count rate decreases. This makes intuitive sense because as the overall distance increases, the rotational change of distance to the source among different points in the detector active volume relative to the overall distance is decreasing. If one were to be interested in measuring the location of a single source, this could be a useful technique to measure the directionality of the source, provided one is close enough

and there is no significant source of other radiation to interfere with the signal directionality.

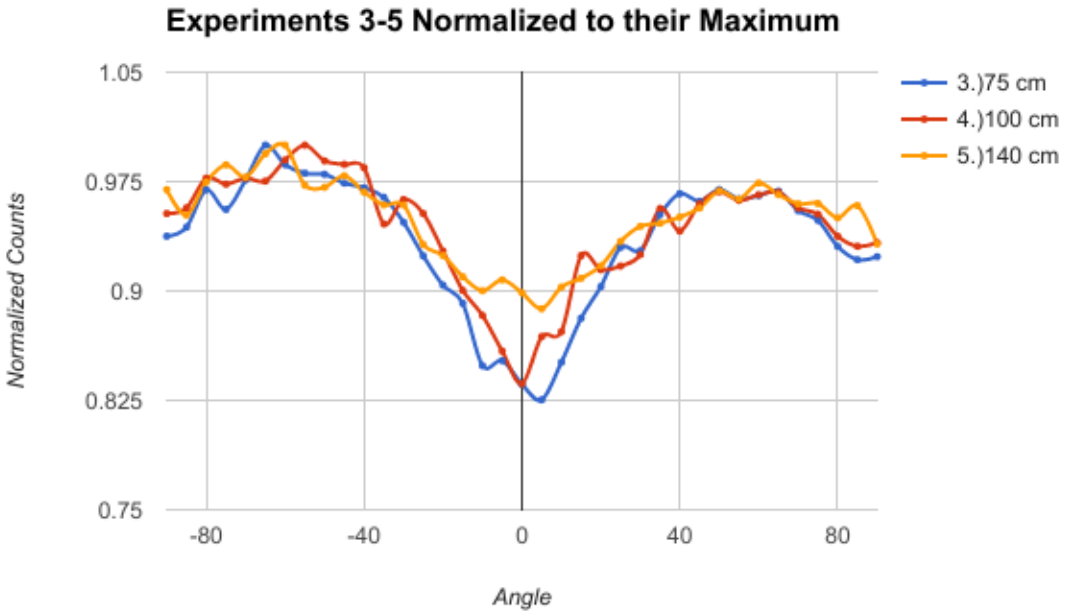


Figure 71. Normalized count rate results for short axis rotation dependence tests at positions 3, 4, and 5.

4.3.4 Short Axis Rotation Dependence with Shielding

The short axis rotation dependence test setup was repeated with variable amounts of shielding. The source was placed in position 4, 100 cm, and rotated 180 degrees about the short axis in increments of 5 degrees. Polyethylene was available in two thicknesses and varieties: 5 cm of 2% borated polyethylene, 10 cm of 2% borated polyethylene, 5 cm of regular polyethylene, and 10 cm of regular polyethylene. For these four new test

parameters, deadtime was less than 0.10% and error was within ±1.7 cps (Figure 72, Figure 73, Figure 74, and Figure 75).

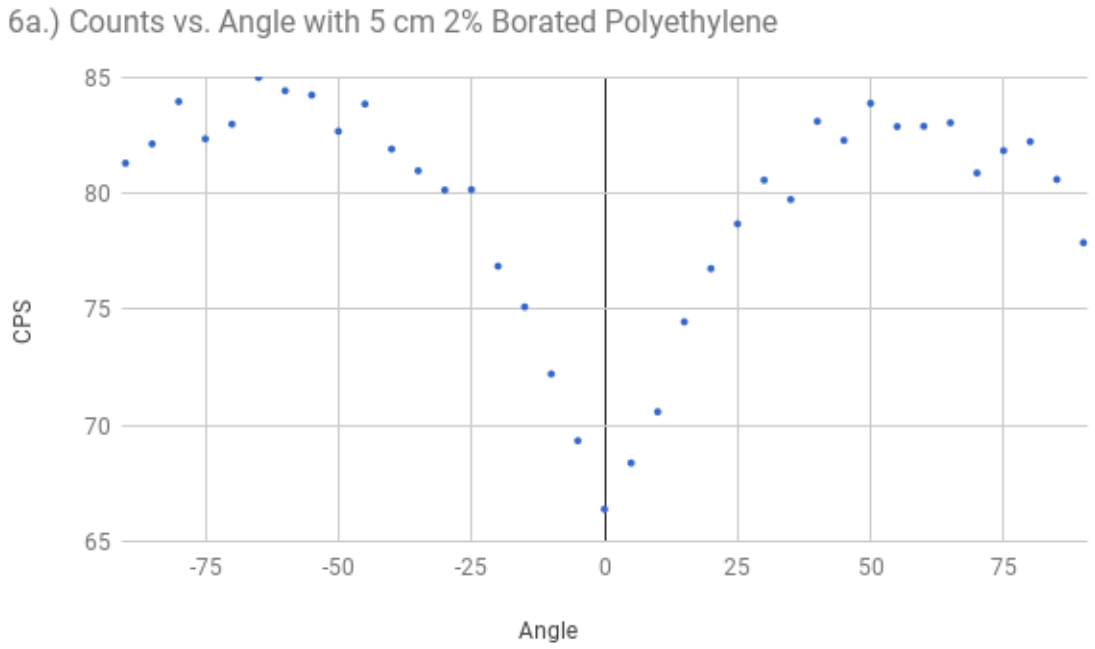


Figure 72. Count rate results for short axis rotation dependence test at position 4 with 5 cm of 2% borated polyethylene.

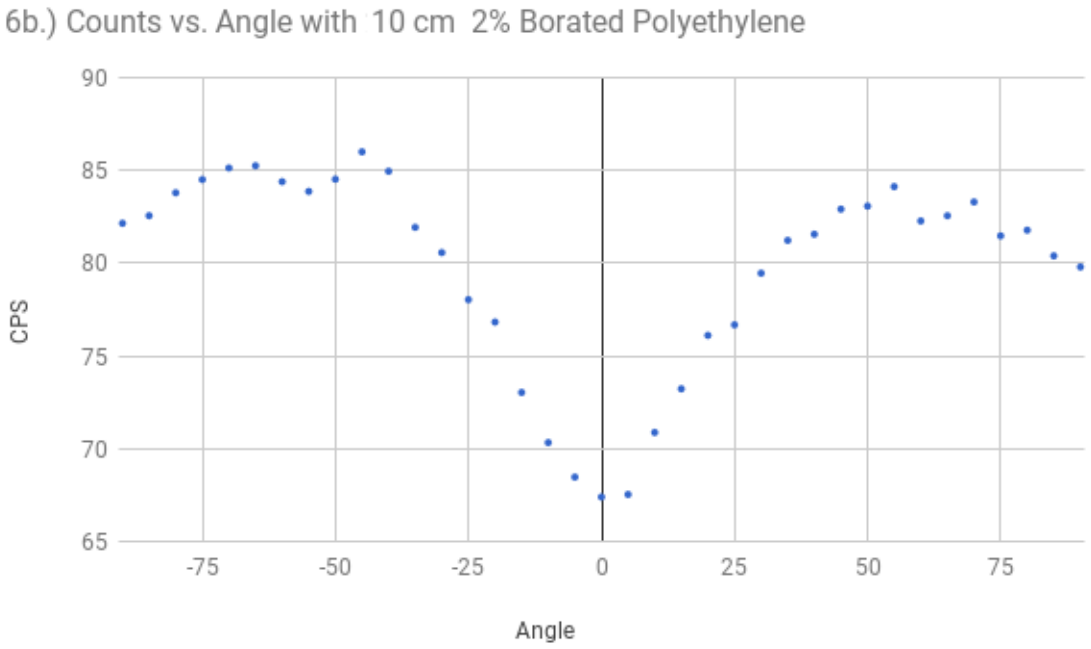


Figure 73. Count rate results for short axis rotation dependence test at position 4 with 10 cm of 2% borated polyethylene.

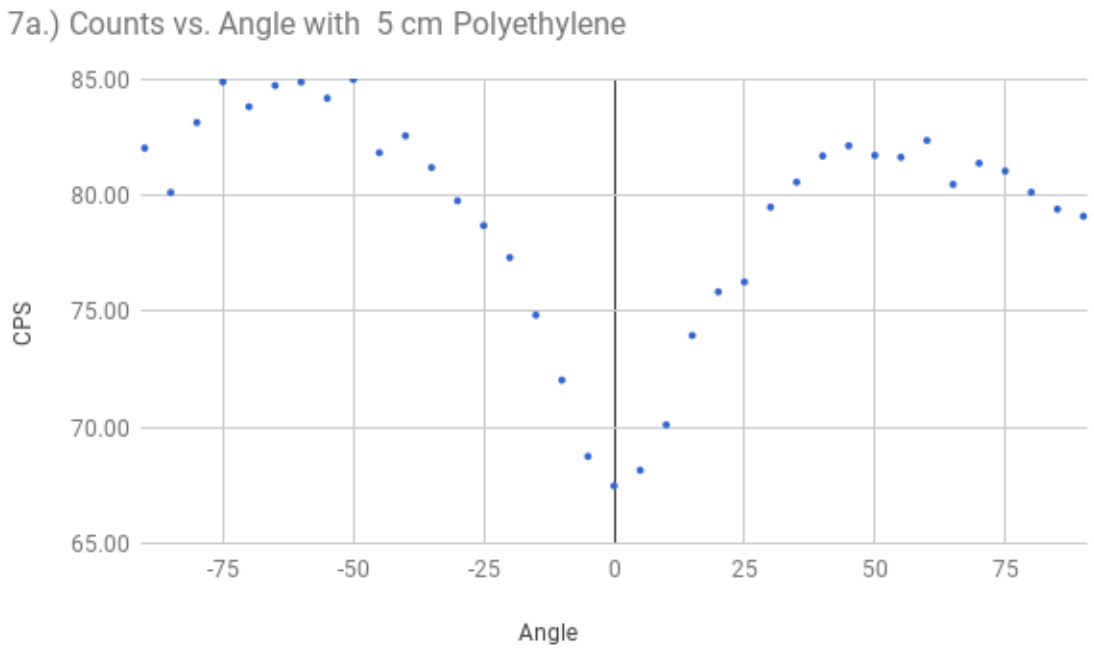


Figure 74. Count rate results for short axis rotation dependence test at position 4 with 5 cm of polyethylene.

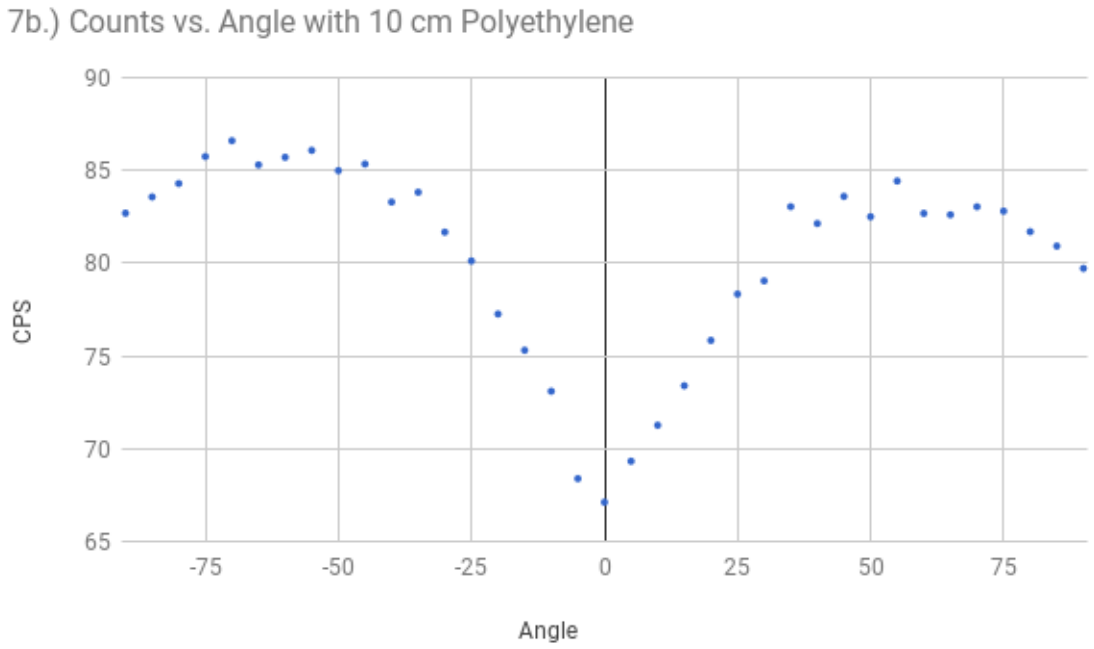


Figure 75. Count rate results for short axis rotation dependence test at position 4 with 10 cm of polyethylene.

Shielding had the opposite effect that distance had on the normalized difference between count rates as the detector was rotated around its short axis; shielding deepened the difference between 0 degrees and other rotational positions (Figure 76). Differences between the thicknesses of shielding were less visible in this experiment. If this short axis rotation dependence were to be used to measure directionality, shielding would likely be advantageous to incorporate into the configuration of the detector.

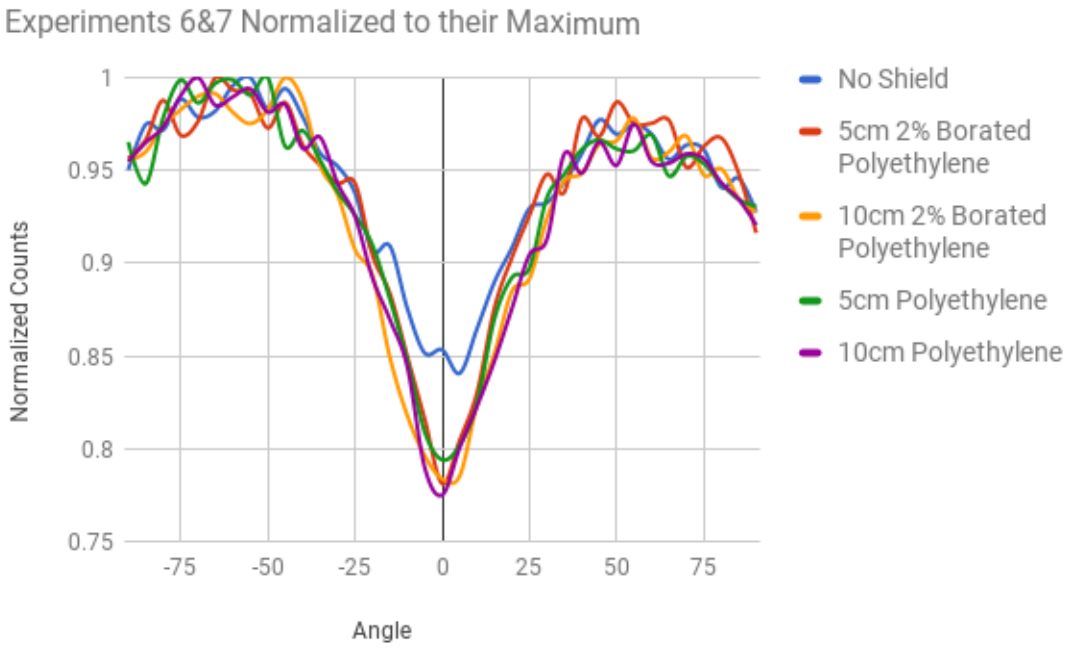


Figure 76. Normalized count rate results for short axis rotation dependence at position 4 with varied shielding configurations.

4.4 MCNP Configurations

Shielding to go around the detector at LANL was simulated and varied to find a configuration that would allow some differences in counts between radiation incoming through an artificial aperture versus other directions. The detector itself was not part of the model. The shielding was simulated and then a surface current tally placed along the interior surface of the shield to count simulated particles that cross through. Notably, it was not treated as required for this work to find a mathematically optimized shielding configuration. The reasoning behind this is twofold. First, the detector shape itself is not considered to be optimized for this application. It was desired to accomplish this work with

an “off the shelf” detector for ease of use and proof of concept for creating shielding directionality. Second, there are cost-benefit considerations regarding shielding and weight capacity of the robot. During MCNP simulations it was readily seen, as would be expected, that increased amounts of shielding increasingly blocked more radiation. However, the size of shielding could quickly become unwieldy. The shielding needed to be thick enough to be able to find a measurable difference between source radiation directions while remaining light enough to be easily carried and mounted upon a robot. The mobile platforms themselves have large payloads, but in a system used to survey an area, it is unlikely that only the neutron detector would be installed (for the Vector platform, a large amount of the payload was already taken up by the mounting plate and robotic arm). To this end, MCNP runs were designed to vary the thicknesses of different layers of shielding.

The basic setup of the MCNP model is an inner void surrounded by a layer of high-density polyethylene (HDPE). HDPE provides the moderating effects of polyethylene while the higher density provides some additional shielding from gamma rays (capture and otherwise). The end caps are treated as voids, and this is to simplify simulation and manufacturing of the test shield; in reality, shielding will need to be placed over these caps if deployed in a location with many sources all around. The overall system is bounded by a sphere of dry air, as would be typical in the American Southwest. Located within the sphere is a Cf-252 neutron source and a diffuse neutron background that can be disabled as desired (Figure 77).

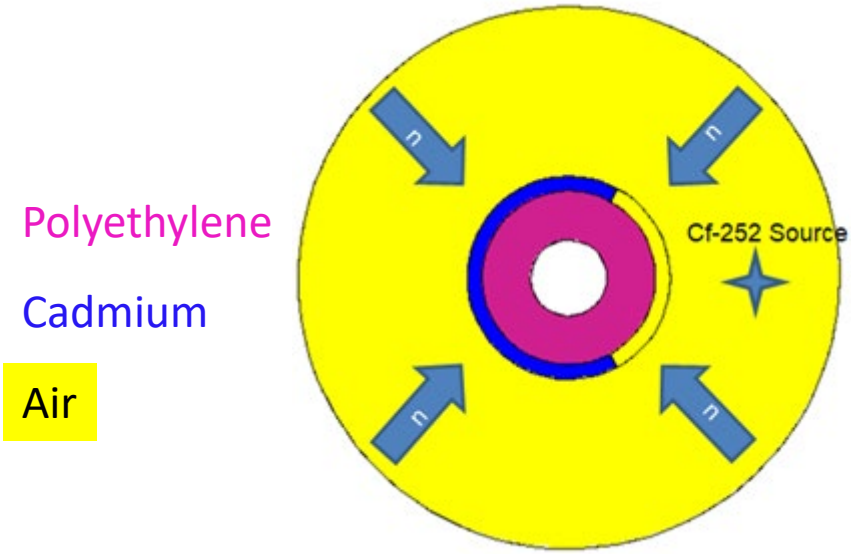


Figure 77. MCNP model of shield, inner void, and two sources (Cf-252 and diffuse incoming background) that can be adjusted.

The initial tests focused on the addition of a variable cadmium lining around the HDPE to attempt an artificial aperture. The thickness of the inner HDPE, the thickness of the cadmium, and the wedge angle were varied to test the results (Figure 78, Figure 79, Figure 80, and Figure 81).

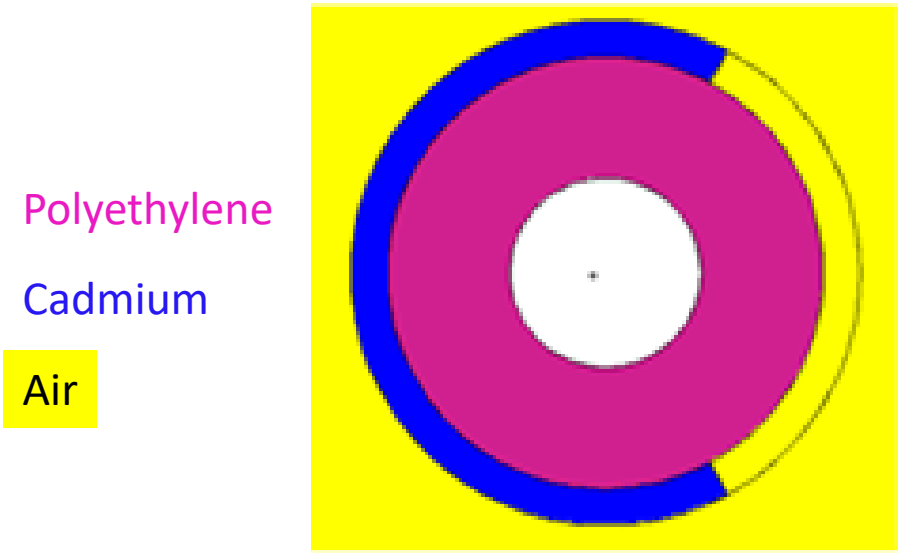


Figure 78. Base example for MCNP model.

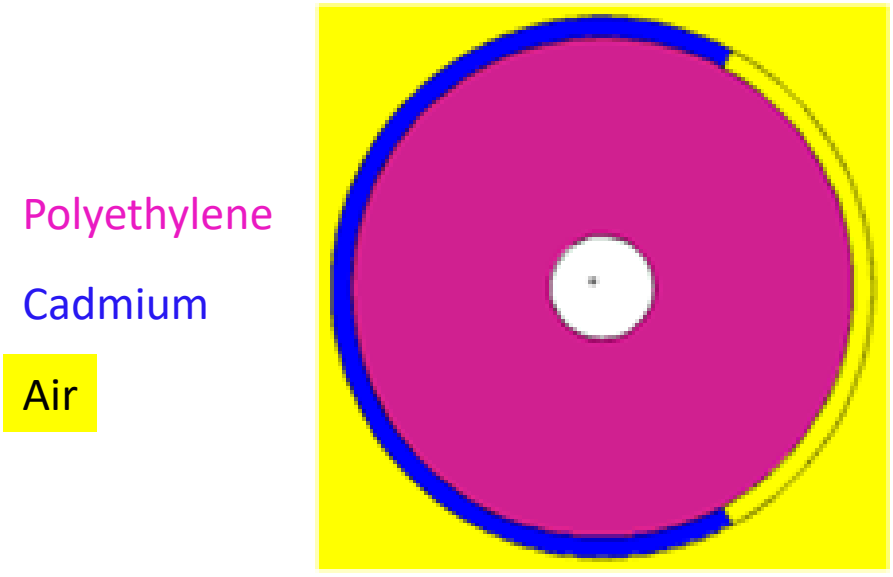


Figure 79. MCNP model with polyethylene thickness adjusted.

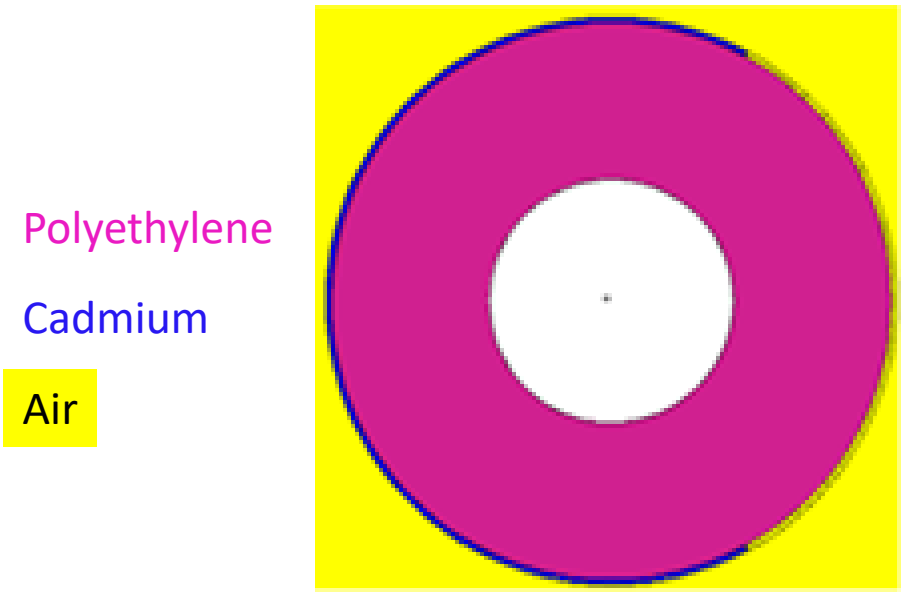


Figure 80. MCNP model with cadmium layer adjusted.

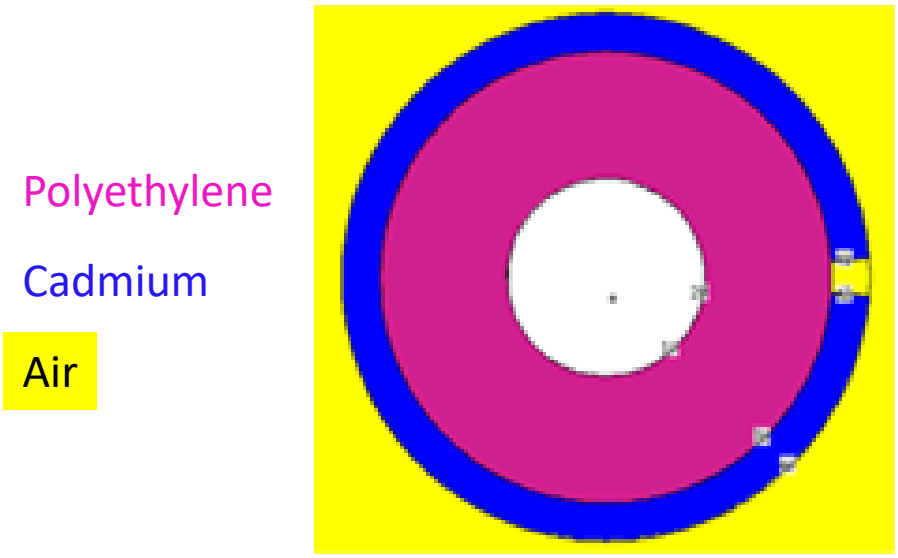


Figure 81. MCNP model with wedge window angle adjusted.

Although this layer provided some additional shielding, which could help filter out background neutrons, it unfortunately did not result in an appreciable difference in directionality. To continue to explore the idea of creating an artificial aperture with shielding, a layer of borated polyethylene was added around the outside of a cadmium layer with a wedge cut out. Small gaps (~0.001 cm) of air can be included within the layers of shielding to create a more realistic model where the air can cause some neutron scattering. As before, the variation in thickness of shielding and size of the wedge were varied. Simulation did not result in significant differences between tallies of neutrons entering the inner void, where the active volume of the neutron detector would be located. The largest difference simulated was approximately 5% in the thermal neutron tally between the source positioned directly over the wedge cutout and the diffuse background. In order to test the shielding, a design was chosen of 2.54 cm inner HDPE layer surrounded by a thin 0.0508 cm stock layer of cadmium and further surrounded by a 2.54 cm layer of borated polyethylene with a wedge of 60 degrees (Figure 82 and Figure 83).

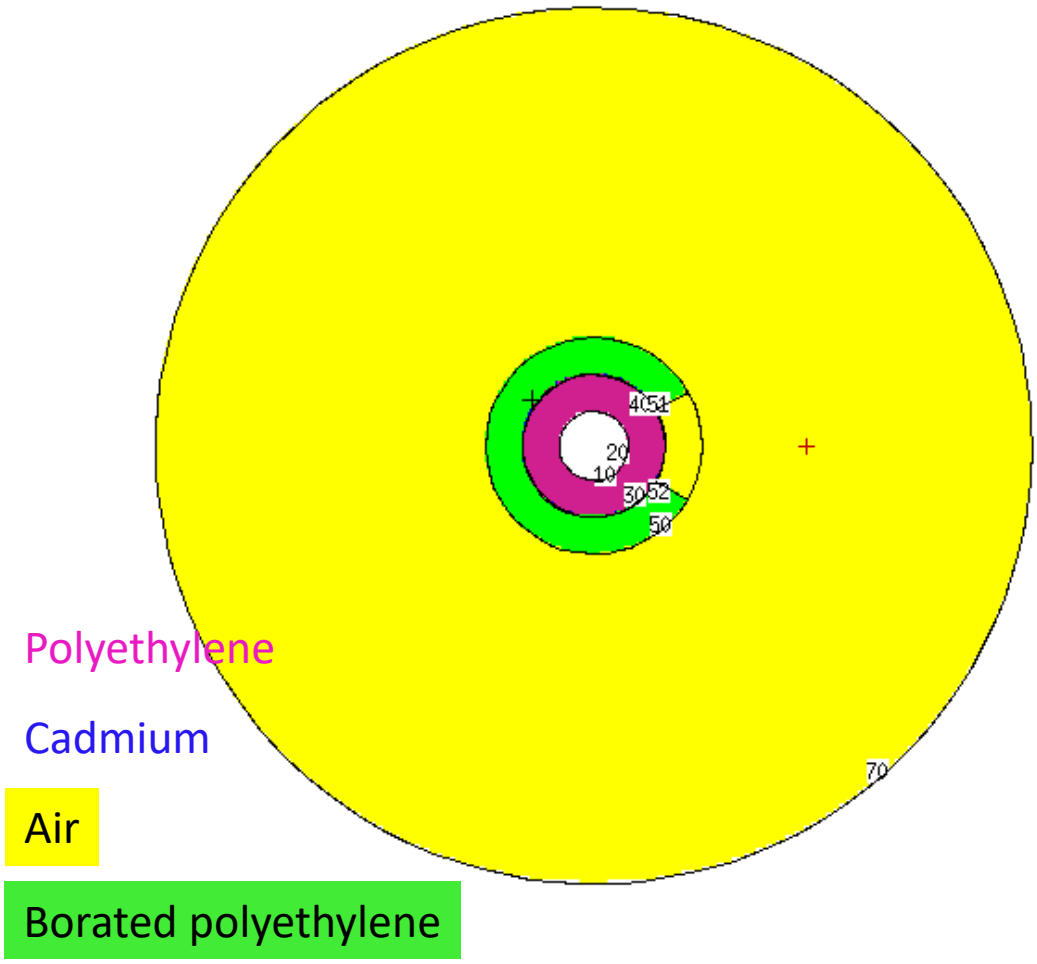


Figure 82. MCNP model with three-layer shield and bounding region.

Figure 83. Close-up view of MCNP model with three-layer shield.

CAD drawings were created in SolidWorks for the manufacture of the shield, and the shielding was fabricated at LANL (Figure 84, Figure 85, Figure 86, and Figure 87).

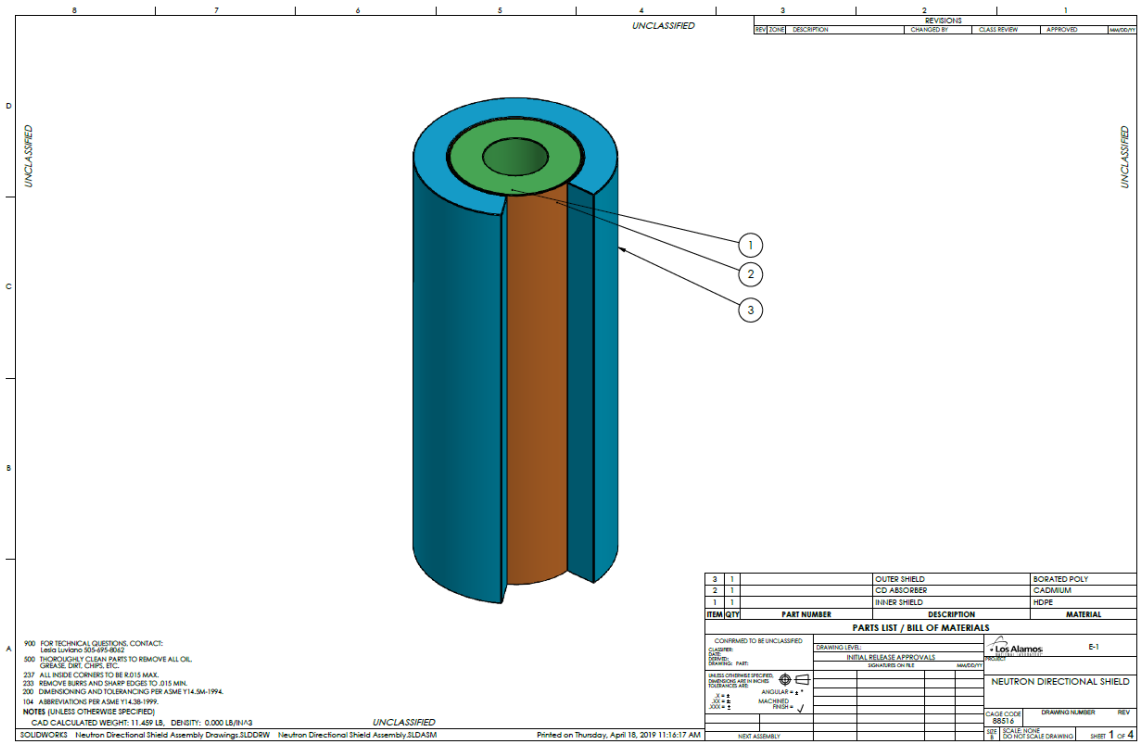


Figure 84. CAD model of three-layer shield.

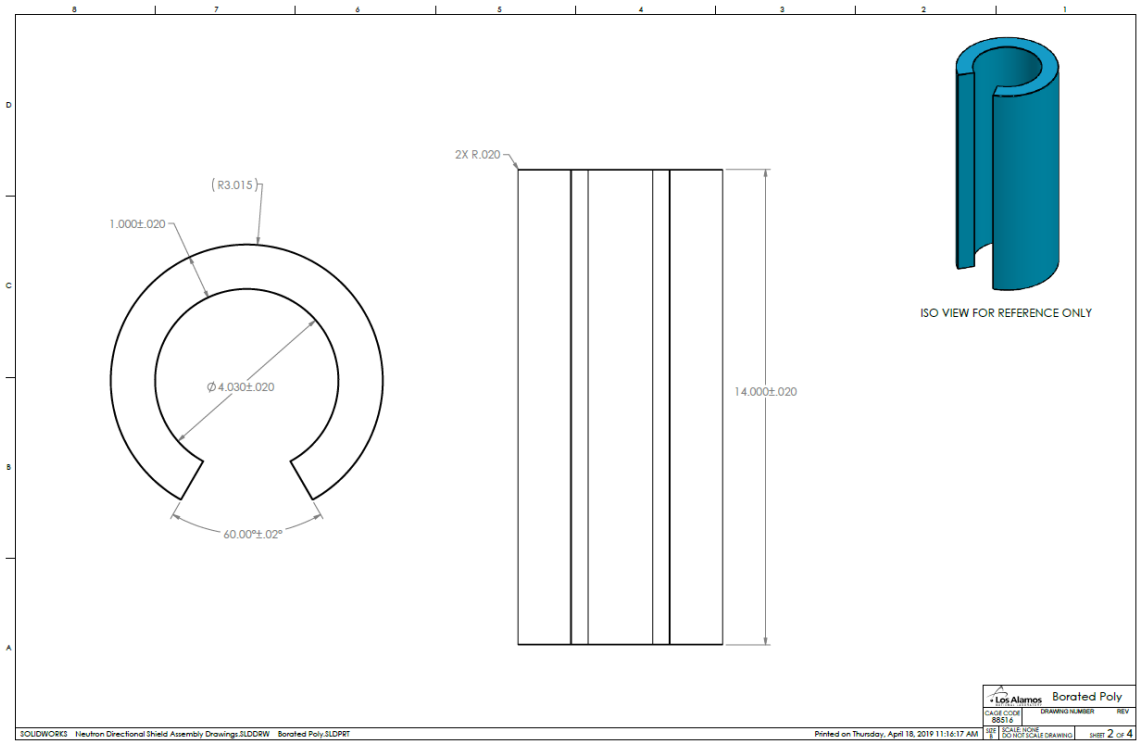


Figure 85. CAD drawing of outer borated polyethylene shield layer.

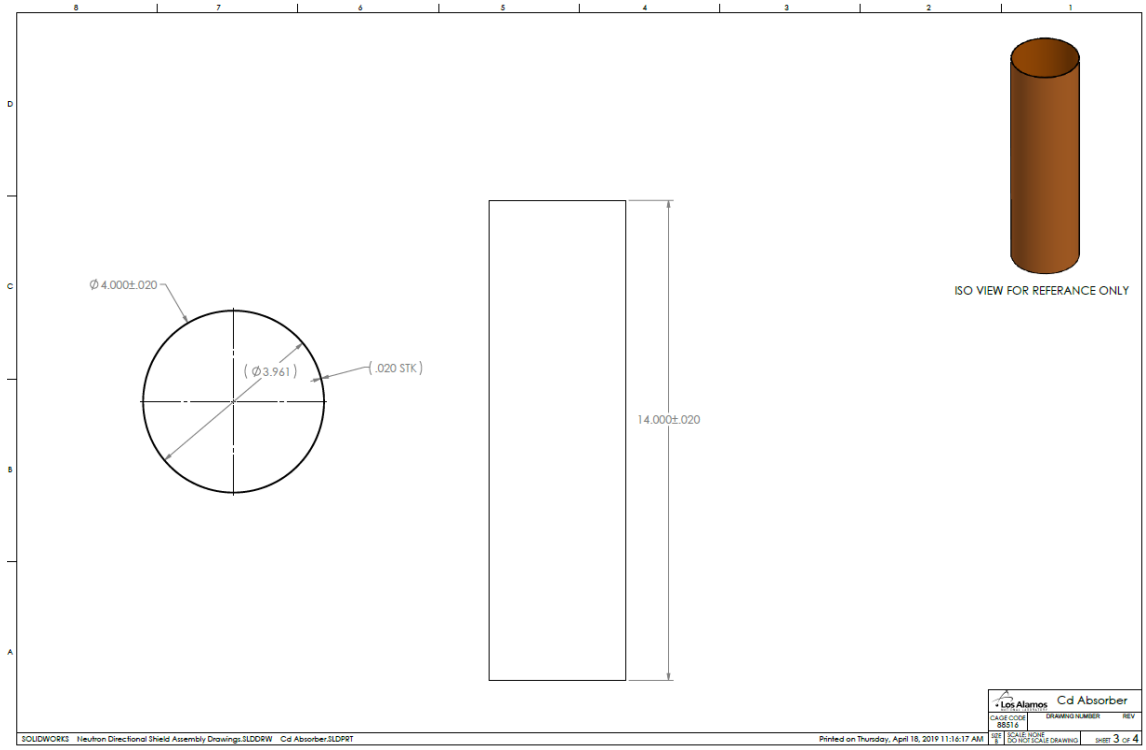


Figure 86. CAD drawing of middle cadmium sheeting layer.

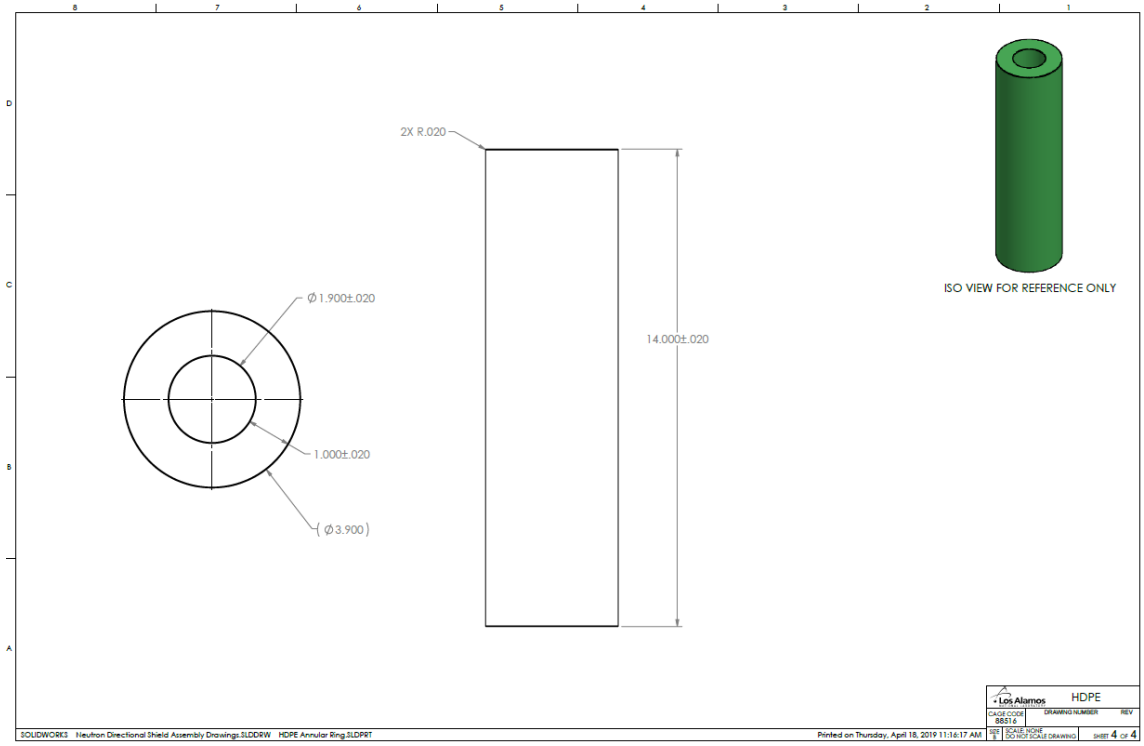


Figure 87. CAD drawing of inner HDPE layer.

4.5 SHIELD CHARACTERIZATION

Once the shield was manufactured, it was taken to TA-35 with the neutron detector to create an experimental apparatus to test the shielding against the MCNP model (Figure 88 and Figure 89). A Cf-252 sealed source was obtained to be used for this purpose. In order to stabilize any scattering interactions with the environment, the detector and shield assembly was rotated instead of moving the source (as was the case with the earlier detector characterization experiments).

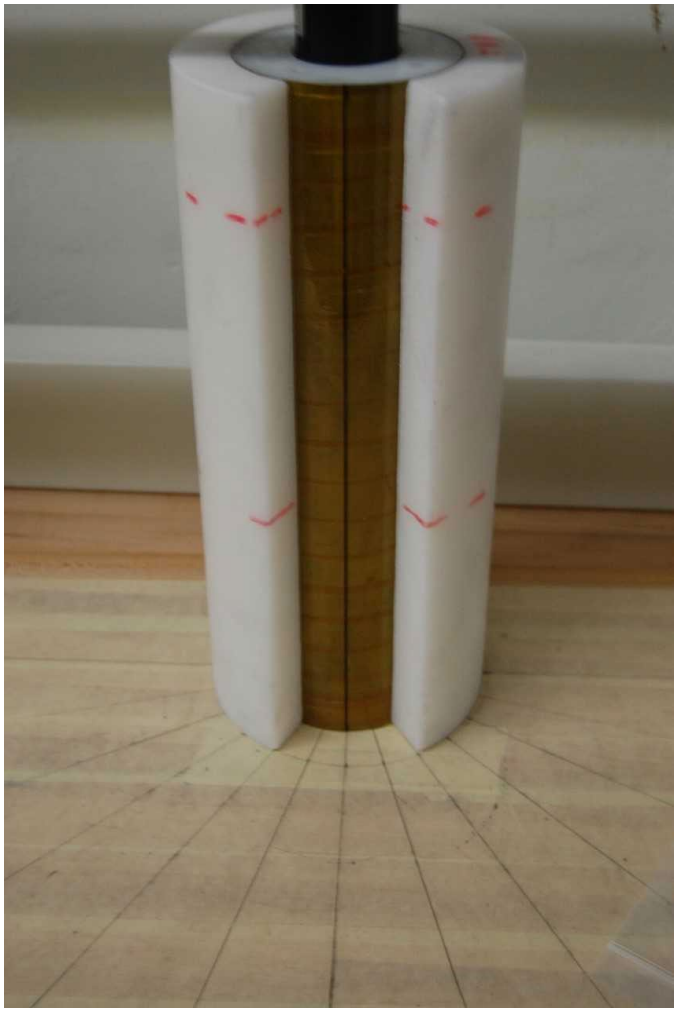


Figure 88. Fabricated neutron shield and detector in test location.

Figure 89. Alternate view of test location with shield, detector, and source stand.

The Cf-252 sealed source is a certified neutron reference standard that was manufactured by Isotope Products Laboratories (located in Valencia, California). The material form is an evaporated metallic salt in a ceramic matrix contained within stainless steel. The active diameter size is 3.17 mm. The initial activity was 1,660 μCi (61,420 kBq) on November 1, 2002. Total uncertainty is listed as 5.8% at a 99% confidence level. Initial flux is listed as 7.12×10^6 neutrons/second. With a half-life of 2.645 years, the flux on the

date of experimental testing was exponentially decayed to 9.4081×10^4 neutrons/second. The source was located 35.56 cm (14 inches) from the detector on a stand to position the source in the middle of the active volume. The 4π area for a source at this distance is 15,890.35 cm^2 ($4\pi R^2$). The fluence is thus the decayed flux over this area or 5.92 neutrons/cm^2-second. Using the 2-D footprint of the active volume (2 inches by 12 inches or 154.84 cm^2), the flux of neutrons from this Cf-252 source incident on the detector active volume can be approximated as 916.74 neutrons/sec. This number is used to calculate the efficiency of detection as the shield and detector were rotated about the long axis. Counts were collected using a custom Python script to communicate with the detector over USB and export the results into a comma-separated values file read with Microsoft Excel. A background count was performed for 30 minutes and found to have a count rate of 5.804 ±0.57 cps with a deadtime of 0.006%. Counts were measured for 5 minutes as the shield was being rotated in increments of 15 degrees around the initial ±90 degrees from the center of the shield window and in increments of 45 degrees around the remaining section (Table 6). The source position varied between the length of the active volume at 0 inches (middle), 6 inches (top), and 12 inches (bottom, Figure 91). Errors for the 6 inch position are shown in Table 6 and Figure 90 whereas errors for the 0 inch position were within ±0.51 counts per second and errors for the 12 inch position were within ±0.47 counts per second.

Table 6. Results from shield rotation test with source at 6 inches.

Detector Rotation (°)	Counts per Second	Absolute Efficiency	Relative Efficiency
0	81.90 ± 0.52	8.30%	100.00%
15	78.94 ± 0.51	7.98%	96.11%
30	76.67 ± 0.51	7.73%	93.13%
45	71.22 ± 0.49	7.14%	85.96%
60	66.99 ± 0.47	6.67%	80.41%
90	67.34 ± 0.47	6.71%	80.87%
135	70.01 ± 0.48	7.00%	84.37%
180	70.53 ± 0.48	7.06%	85.06%
225	70.62 ± 0.49	7.07%	85.18%
270	65.35 ± 0.47	6.50%	78.25%
315	69.83 ± 0.48	6.98%	84.14%
330	75.33 ± 0.50	7.58%	91.37%
345	80.66 ± 0.52	8.17%	98.37%
360	81.35 ± 0.52	8.24%	99.27%

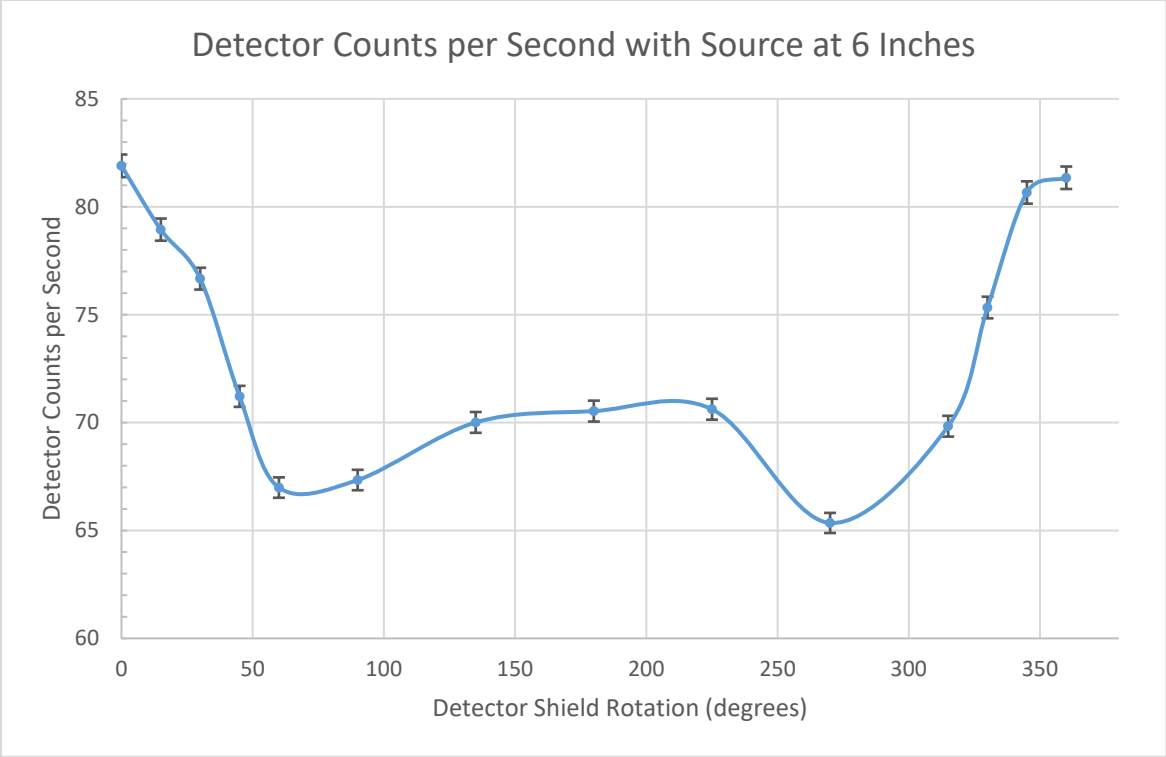


Figure 90. Detector count rate with source at 6 inches and associated count rate errors.

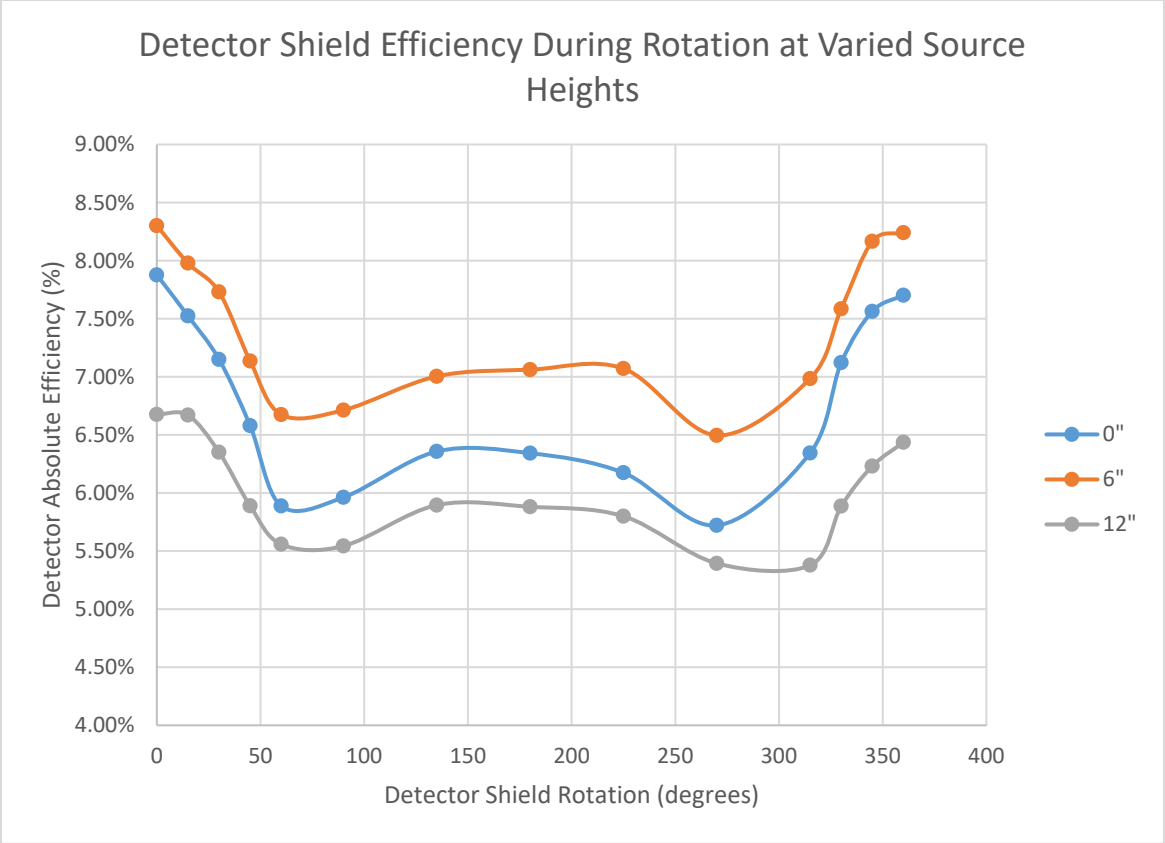


Figure 91. Variations in detection efficiency with varied source heights.

The shield resulted in differences within a few percentage points for the absolute efficiency in detection. The directionality is more clearly shown in the relative efficiencies with a maximum difference of 21.75%. There was an interesting increase in efficiency when the source was 180 degrees rotated versus the positions of 90 degrees and 270 degrees. This change in the efficiency is theorized to be from increased leakage *out* through the window of shielding at ±90 degrees while neutrons are more likely to be scattered back into the active volume than leak when at 180 degrees.

4.6 INTEGRATED DEMONSTRATION

The full system was integrated on the Vector platform to be able to autonomously navigate the corridor at TA-35 and proceed to several inspection points while collecting data. The test corridor was scanned using LiDAR on the Vector platform and created into a map using the built-in Vector Dispatcher mapping software. Waypoints were created within this map to position the platform in front of and orient it directly towards container storage locations setup for this test. This demonstration replicated the waypoint navigation used previously on the Pioneer LX. The Vector platform navigated autonomously between set waypoints using the elastic band planner from ROS as integrated into the Vector platform control firmware by Waypoint Robotics. Higher level control of the navigation system was accomplished via a custom Python program implementing WebSocket communication directives. WebSocket is a two-way communication protocol layered over TCP (Transmission Control Protocol) connections (Fette, Google, Inc., Melnikov, & Isolde Ltd., 2011). This program sent commands to the Vector platform to control when to navigate to a new waypoint, verified that the platform was at the correct waypoint via platform-reported position checking, and integrated control of the UR5e arm and neutron detector. While the UR5e arm has several safety features previously discussed, the waypoint position checking was implemented for safety so as to only execute arm movement to inspect the containers when at the appropriate position. Initial position and orientation of the platform within the mapped room is selected within the Dispatcher software.

Several cabinets were set up and stocked with storage containers (Hagan, SAVY-4000, and slip top metal types) and drums (5 gallon and 55 gallon sizes, Figure 92, Figure 93, Figure 94, Figure 95, and Figure 96) to mimic a SNM storage environment. Some of the containers were damaged as part of the auxiliary test involving the cameras and arm

mounted on the Vector. The damage present was in the form of dents, broken sealing tape, or all-purpose flour placed on the lid to mimic the “white powder” issues previously mentioned in Chapter 1 (as flour is relatively benign and did not require any changes to safety or security plans). Some sealed sources were placed in various containers throughout the area. The majority of these sources were Pu-238 heat-source standards, and thus not great sources of neutrons. A Pu-239 source (<4% Pu-240) was placed in a container on one of the shelves above the plane of detector mounting on the robot. On the second cabinet, the Cf-252 source previously used was placed in a container on one of the shelves closer to the plane of the detector mounting on the robot. The scan started at an initial home position and navigated through waypoints set at storage cabinet 1 (A), drum 1 (B), cart 1 (C), storage cabinet 2 (D), drum 2 (E), and then returned to the initial home position. The scan positions are shown in x and the notional path in x. As the scan progressed, the neutron count rate noticeably increased during approach to the cabinets and reached a maximum when the detector was directly facing the Cf-252 source; it began to decrease again as the platform turned and navigated away (Figure 99).

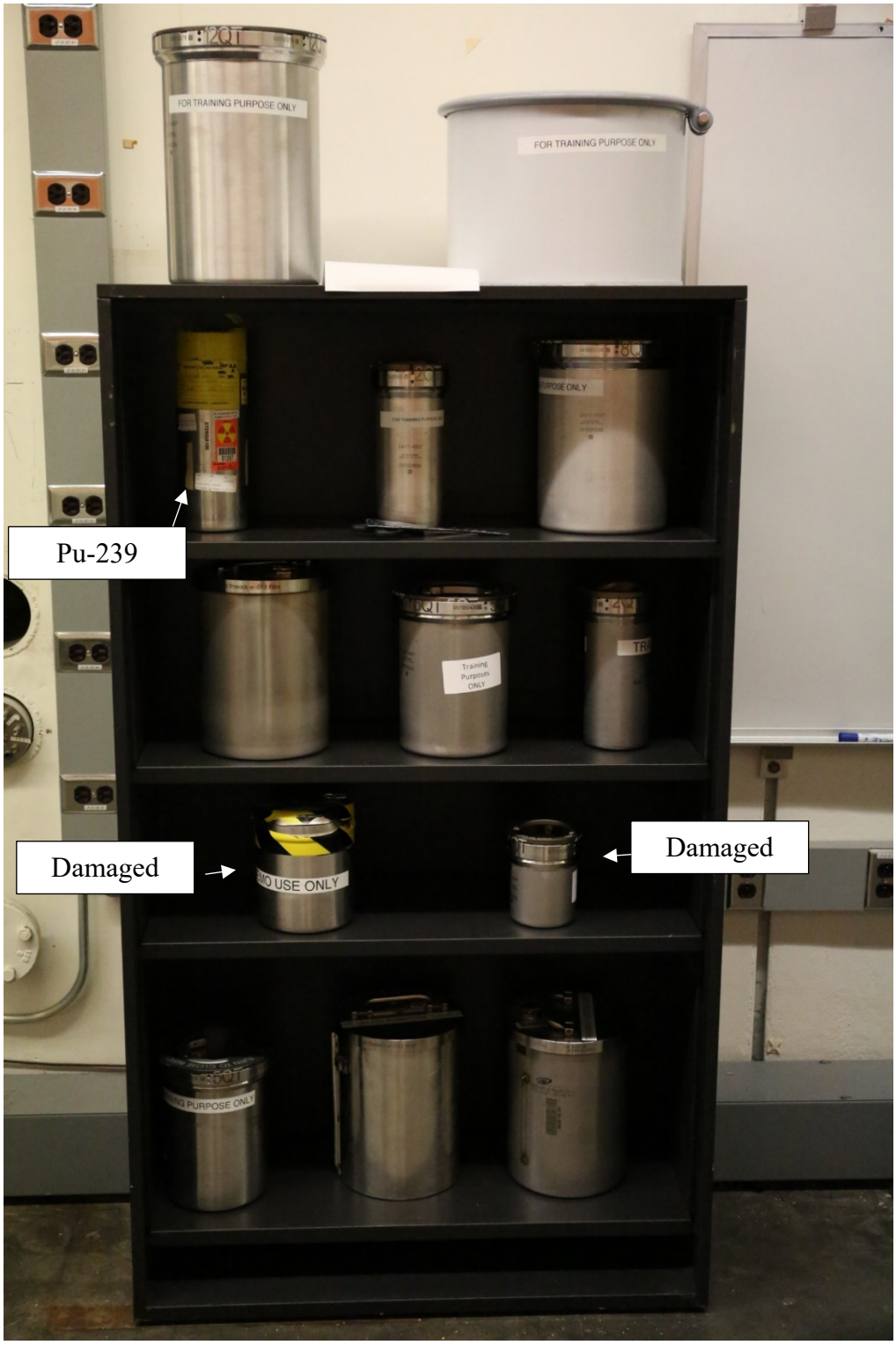


Figure 92. Storage cabinet 1 of integrated system test.

Figure 93. Storage cabinet 2 of integrated system test.

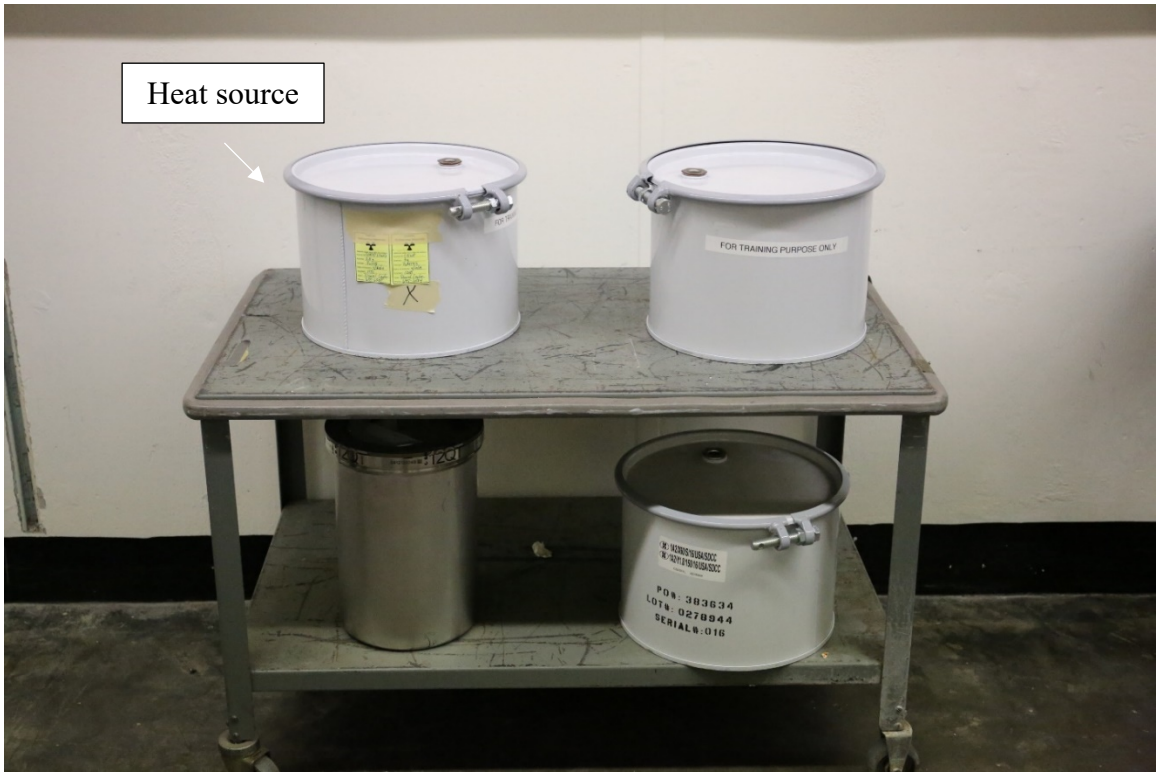


Figure 94. Storage cart 1 of integrated system test.

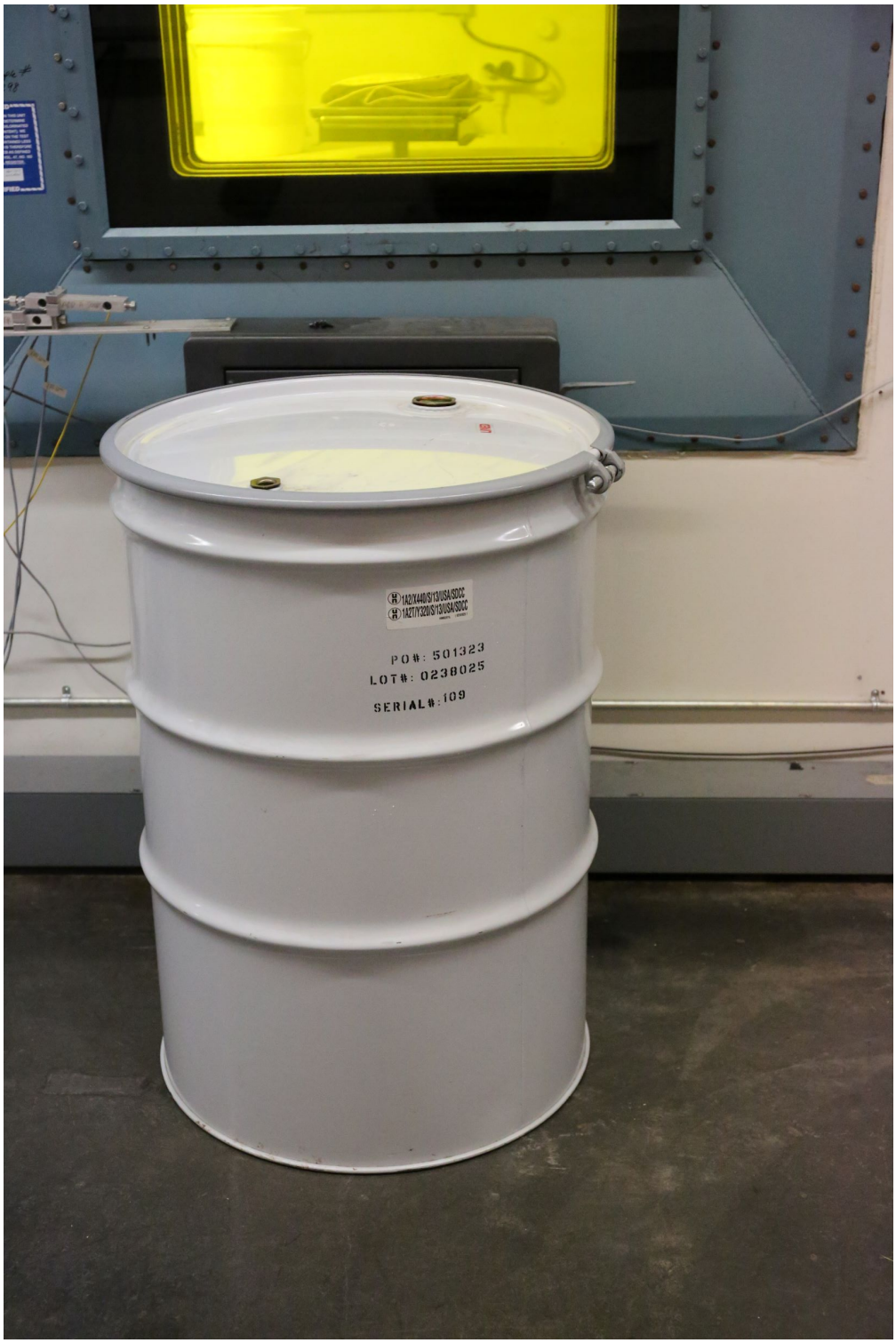


Figure 95. Drum 1 of integrated storage test.

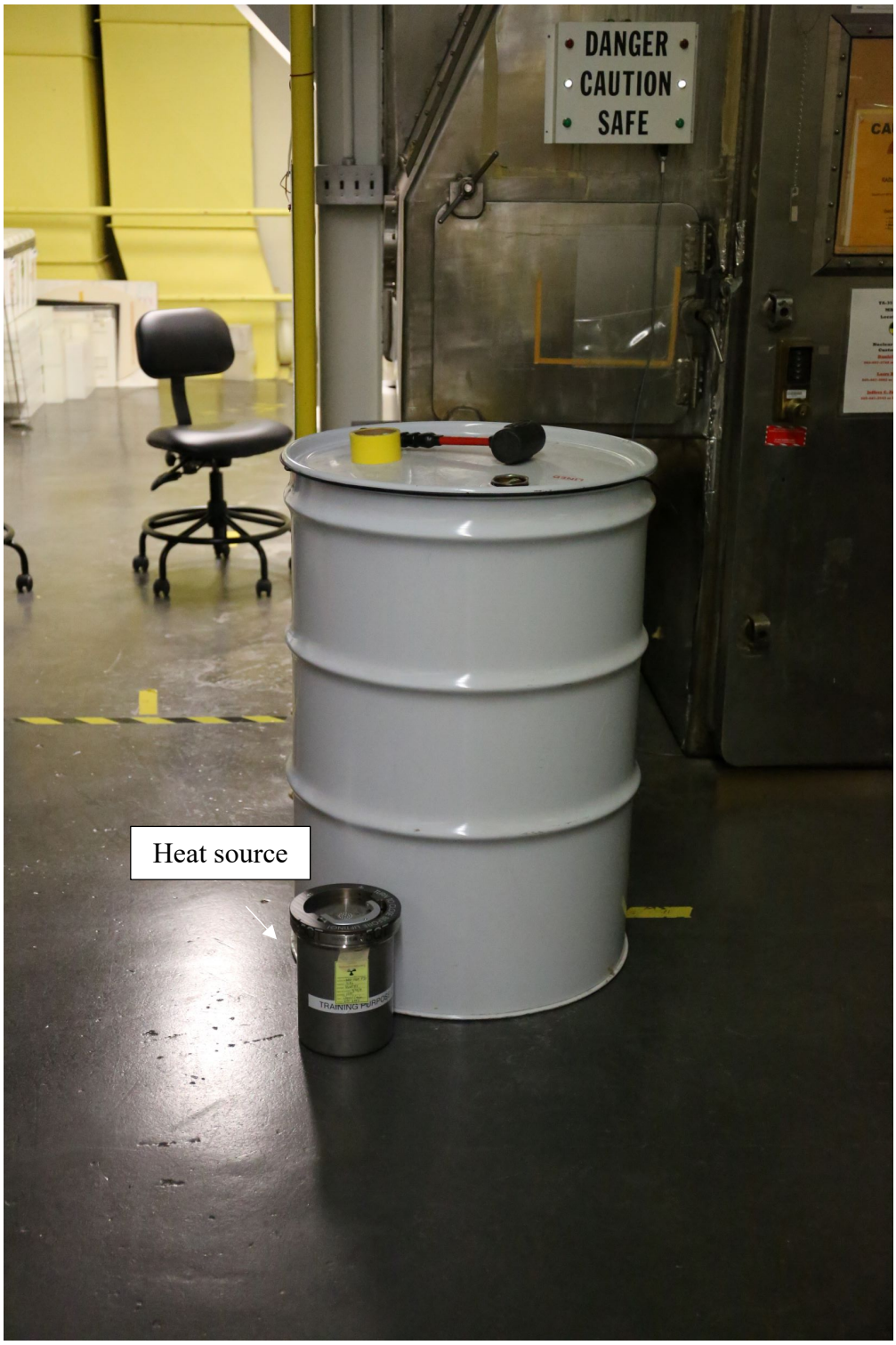


Figure 96. Drum 2 of integrated system test.

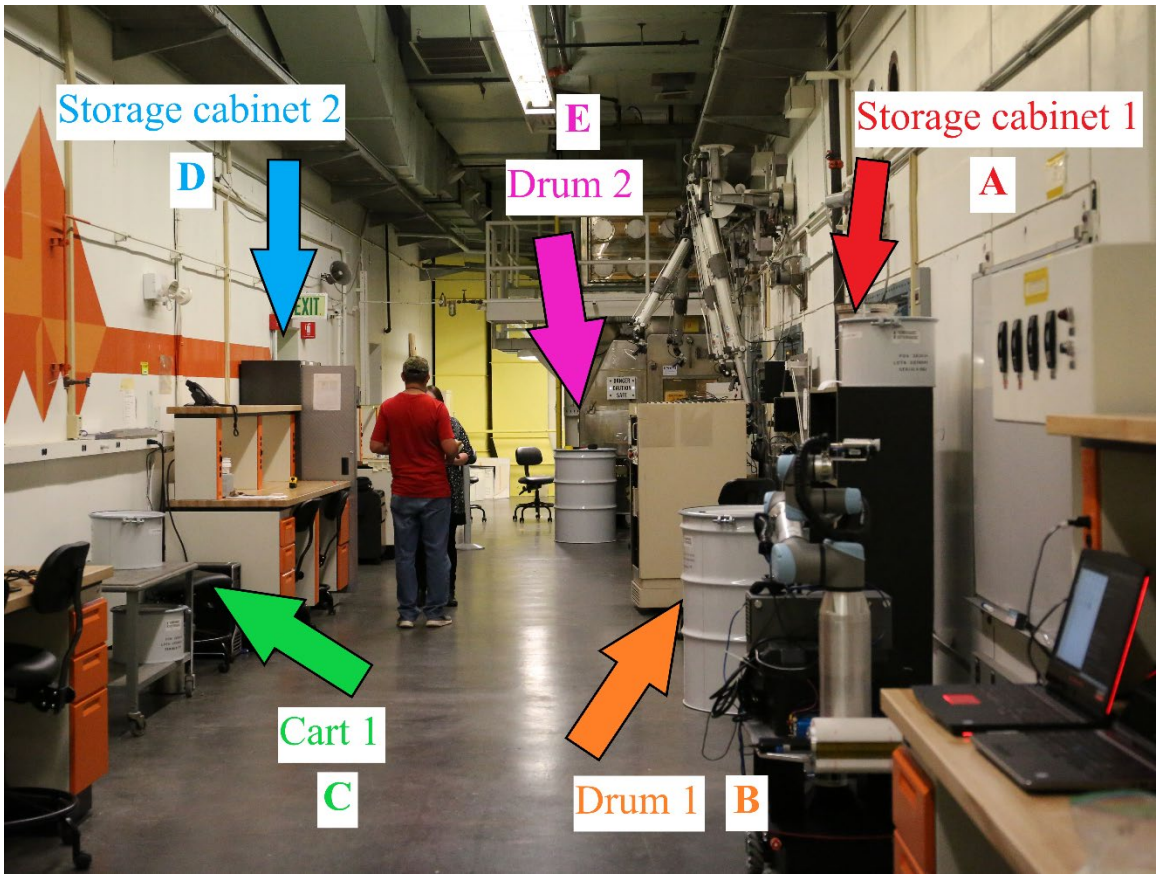


Figure 97. The integrated system test scanning locations of A) Storage cabinet 1 (red), B) Drum 1 (orange), C) Cart 1 (green), D) Storage cabinet 2 (blue), and E) Drum 2 (pink).

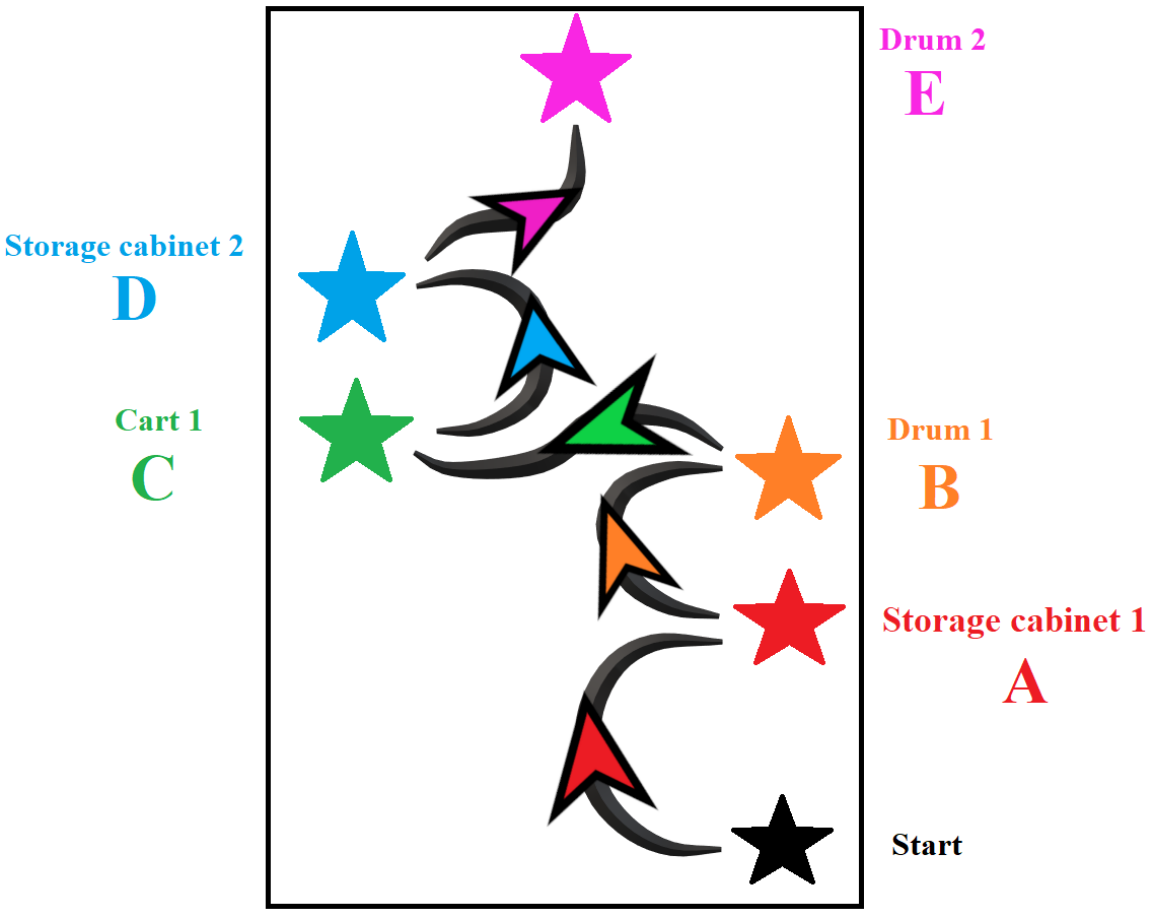


Figure 98. Notional path traveled (not to scale) for integrated system test with locations A) Storage cabinet 1 (red), B) Drum 1 (orange, C) Cart 1 (green), D) Storage cabinet 2 (blue), and E) Drum 2 (pink).

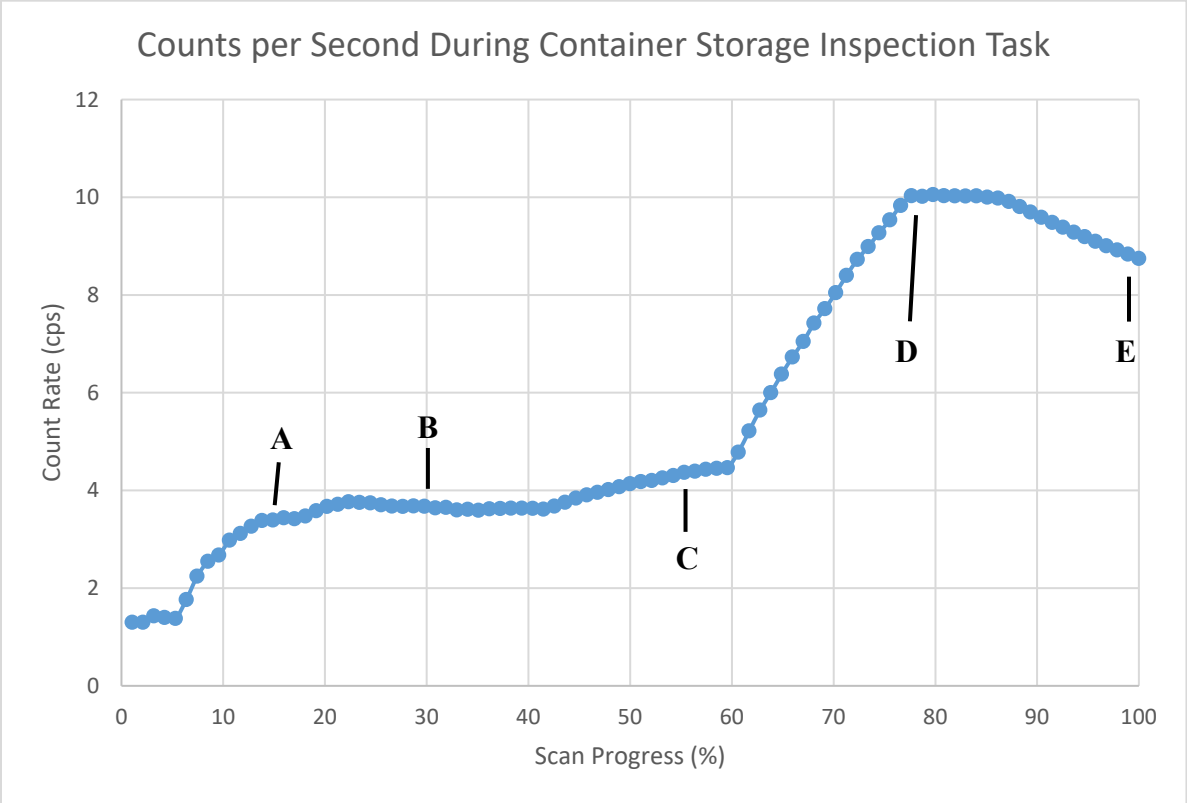


Figure 99. Neutron count rate during integrated system test with room positions indicated.

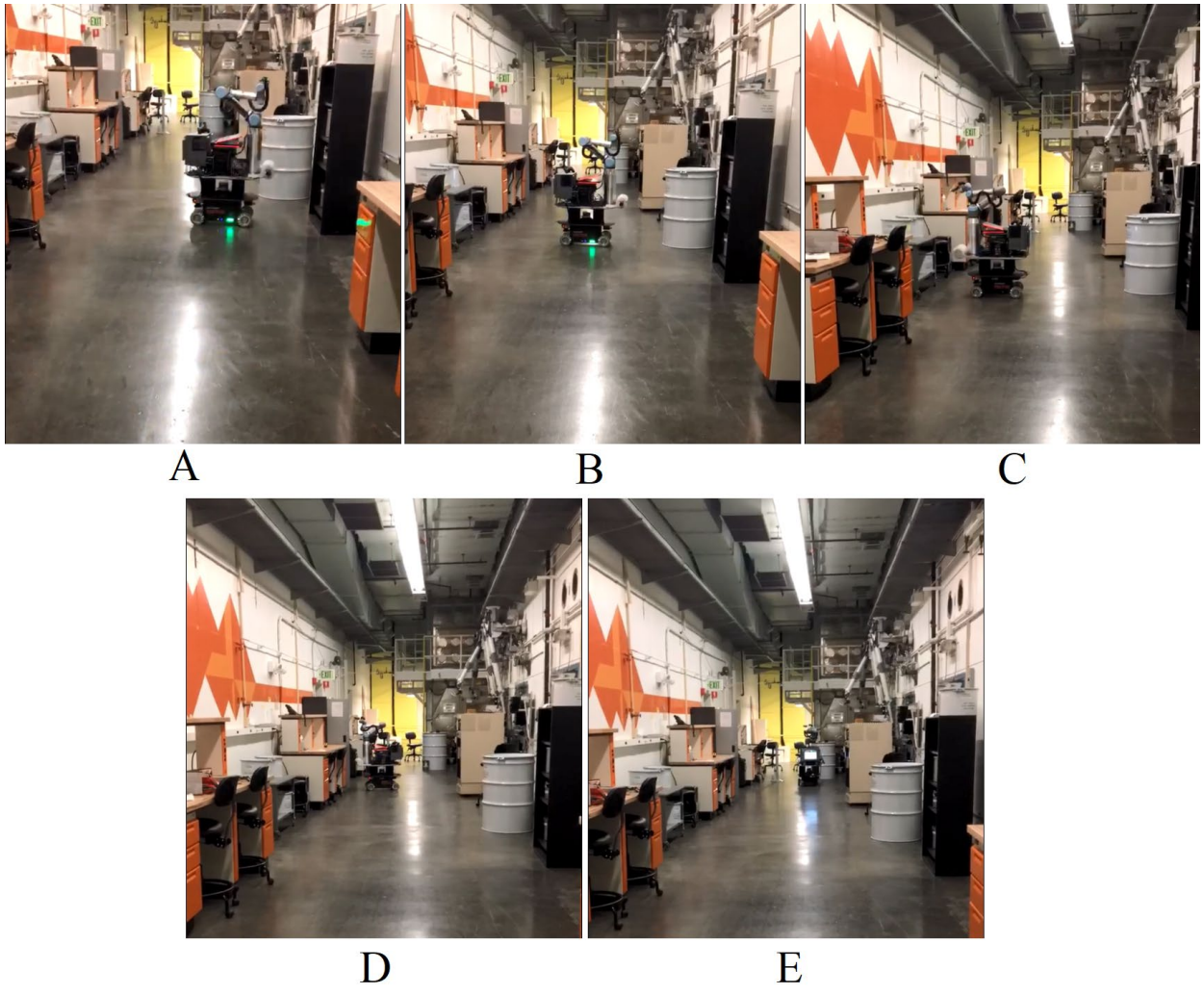


Figure 100. Scanning room at locations A) Storage cabinet 1, B) Drum 1, C) Cart 1, D) Storage cabinet 2, and E) Drum 2.

The container inspection portion of the test mainly sought to confirm that issues with the containers would be able to be identified by a human operator later reviewing the data. The collection of camera data functioned as an initial proof of concept for automated container surveillance. When the platform confirmed it was at the correct location (locations A through E in Figure 100), the Python program then sent a message over the internal platform network to the UR5e arm controller to load and execute the corresponding

scanning motion for the arm to orient and move the dual cameras to inspect the containers. Similar to how the platform navigated, the arm motions were preprogrammed waypoints that the arm controller then calculated and executed motion between. A view of the drum 2 location while the base platform is located at drum 1 is shown in Figure 101. While the platform was navigating between locations, the UR5e arm was placed in a stowed configuration for safety (Figure 43).

Figure 101. Image collected from visual camera pointing towards drum 2 while platform is located at drum 1.

The initial test runs of the integrated system used the visual camera as-is (example images of containers shown in Figure 102, Figure 103, and Figure 104). Later test runs affixed a Streamlight ProTac 1L 350 lumen flashlight along the line of sight of the cameras to get a qualitative idea of the effects of increased illumination on the camera (example images of containers shown in Figure 105, Figure 106, Figure 107, and Figure 108). In all ten test runs with the cameras, 100% of containers that were damaged were able to be easily identified upon human review. The addition of the flashlight caused some specular reflection from the stainless steel cans which could have the potential to interfere with inspection, but was deemed overall helpful in visualizing containers which may have shadows cast on them from the shelving they rest on (compare Figure 105 to the shadows seen in Figure 102).

Figure 102. Visual camera image of container with Cf-252 source in storage cabinet 2.

Figure 103. Visual camera image of container with broken sealing tape in storage cabinet 1.

Figure 104. Visual camera image of left side of bottom row of storage cabinet 2.

Figure 105. Visual camera image of container with simulated "white powder" on lid in storage cabinet 1 with flashlight attached.

Figure 106. Visual camera image of container with broken sealing tape and SAVY-4000 container below it in storage cabinet 1 with flashlight attached.

Figure 107. Visual camera image of containers in storage cabinet 2 with flashlight during execution of base platform movement to storage cabinet 2.

Figure 108. Visual camera image of dented container in storage cabinet 2 with flashlight.

The thermal camera testing provided some interesting results. Baseline examples of the thermal environment of the testing area and empty containers without any

radiological sources present are shown in Figure 109 and Figure 110 using common thermogram color scale conventions.

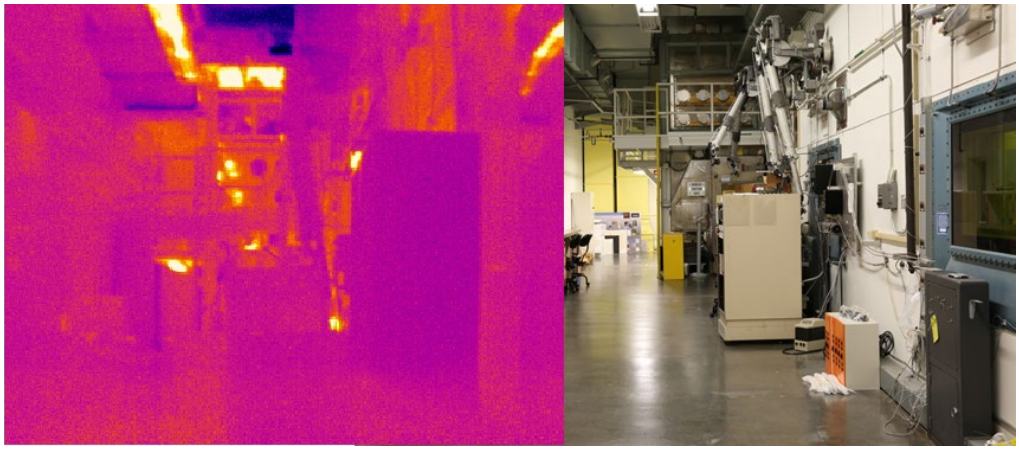


Figure 109. Thermal camera image (left) with related photo (right) of view down testing hallway at TA-35.

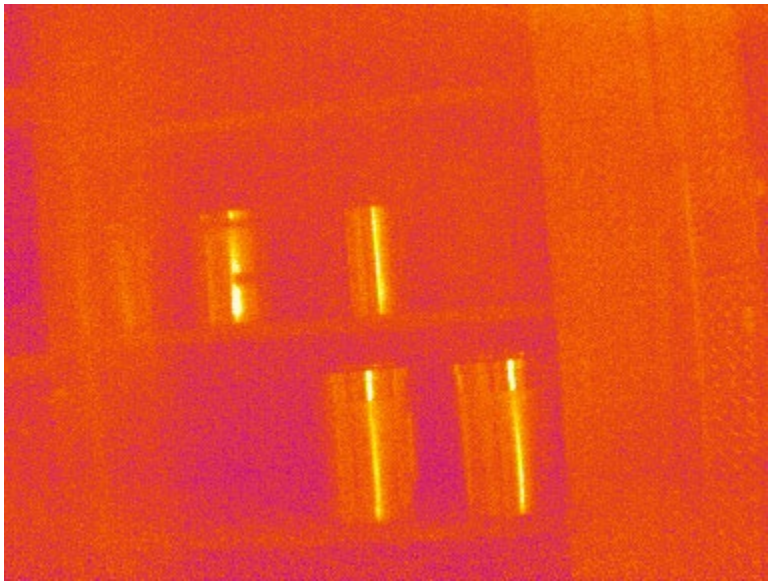


Figure 110. Thermal image of empty cans in storage cabinet 2.

As to be expected, the Pu-239 and Cf-252 sources are not thermally hot like the Pu-238 sources which can be readily seen in the relevant thermal images (Figure 111 and Figure 112).

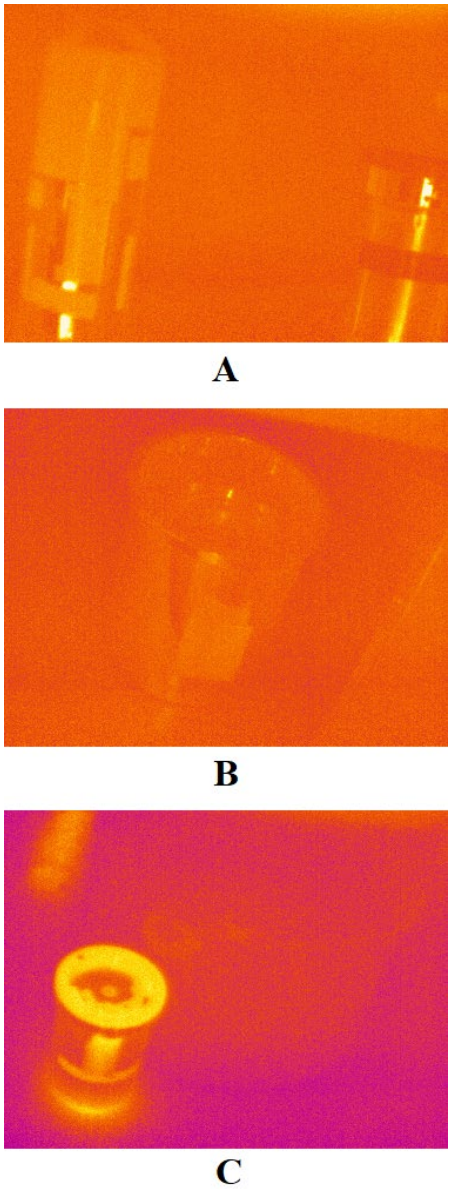


Figure 111. Thermal camera image of A) Pu-239 source (left) next to empty container (right) in storage cabinet 1, B) Cf-252 source in storage cabinet 2, and C) Pu-238 heat source next to drum 2.

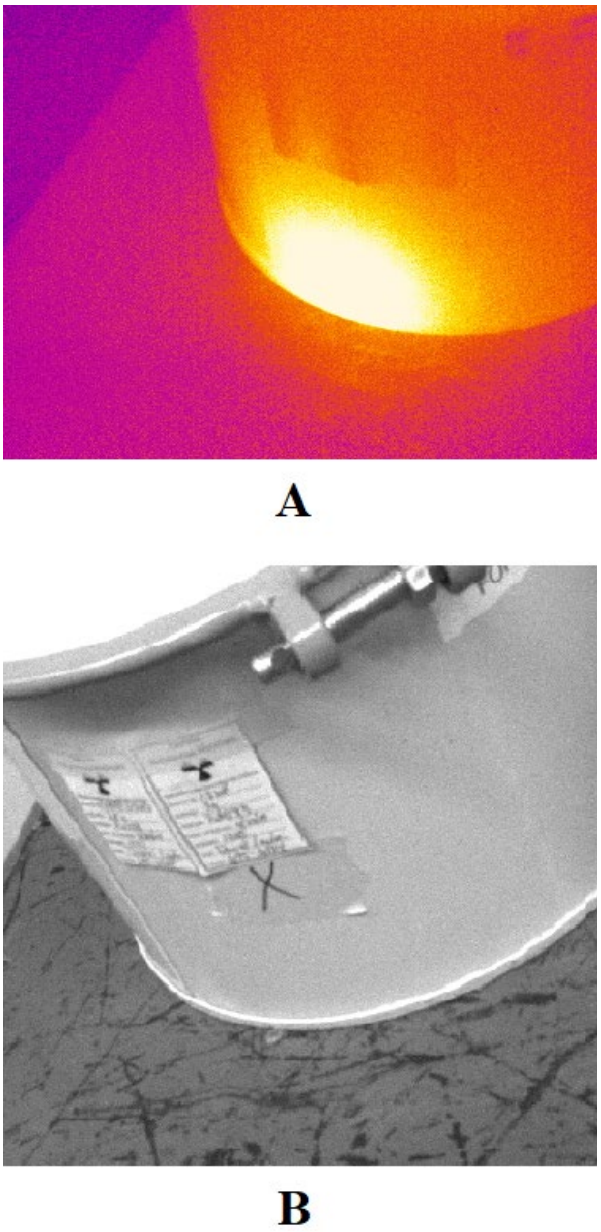


Figure 112. A) Thermal camera image of Pu-238 source in 5 gallon container on cart 1 and B) Visual camera image of the same source and container.

Using the Optris PIX Connect software (release version 3.3.3027.0) included with the thermal camera, the temperatures within a thermal image can be analyzed. For example,

the peak temperature in the image of the 5 gallon container on cart 1 shown in Figure 112 was 78.4 degrees Fahrenheit. The mean temperature in Figure 112 was 72.7 degrees Fahrenheit with a standard deviation of ±1.2 degrees Fahrenheit. A histogram of the temperatures measured in Figure 112 is shown in Figure 113. The color scale represents a temperature range within ± 3 standard deviations from the mean.

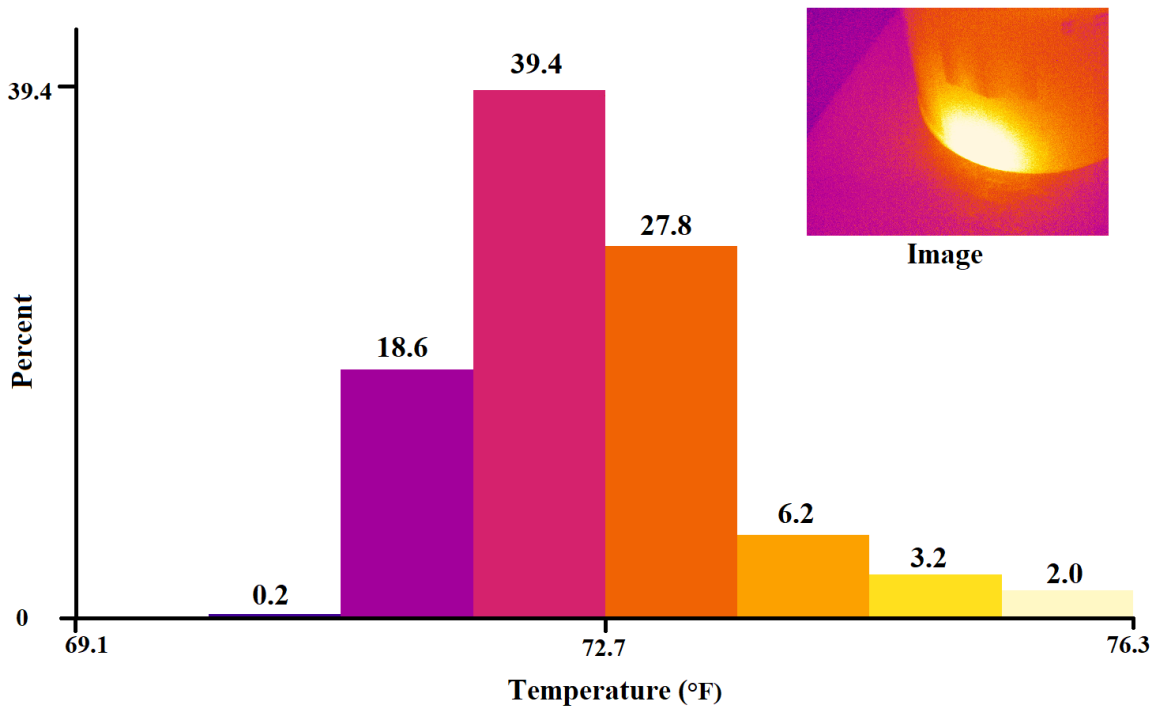


Figure 113. Histogram of temperatures measured from thermal camera image of Pu-238 source in 5 gallon container on cart 1.

As it is electrically powered, the robot itself generates heat that can be seen in thermal reflections off the containers. However, the Pu-238 source located in storage cabinet 2 aptly demonstrated the complications that could potentially occur with thermal reflections. A thermal camera image taken while the platform is executing a move from cart 1 to storage cabinet 2 is shown in Figure 114. The leftmost dented container is next to

the Pu-238 source container which is brightly glowing with heat especially visible on the base, lid, and information labels affixed to the surface of the container.

Figure 114. Thermal camera image of dented and Pu-238 containers in storage cabinet 2 during execution of base platform movement to storage cabinet 2.

Storage cabinet 2 has metal diamond plate doors that were open during testing (the right door of storage cabinet 2 is visible in Figure 93). As the thermal camera is recording during the UR5e arm movement to inspect the containers, the left diamond plate door is visible and shows a reflection of the Pu-238 container located within the cabinet. Figure 115 represents successive images taken by the thermal camera of storage cabinet 2 as the arm pans to the right. The reflection can be seen within the diamond plate pattern on the door. Additionally, thermal reflections such as those visible on the dented container could mistakenly be believed to originate within that container. Since all collected images in this test were reviewed by humans familiar with the environment, it was easy to determine which parts of the images were from the directly observed heat source containers and which

were reflections. However, this could likely be confusing to a computer if analysis of images is automated in the future.

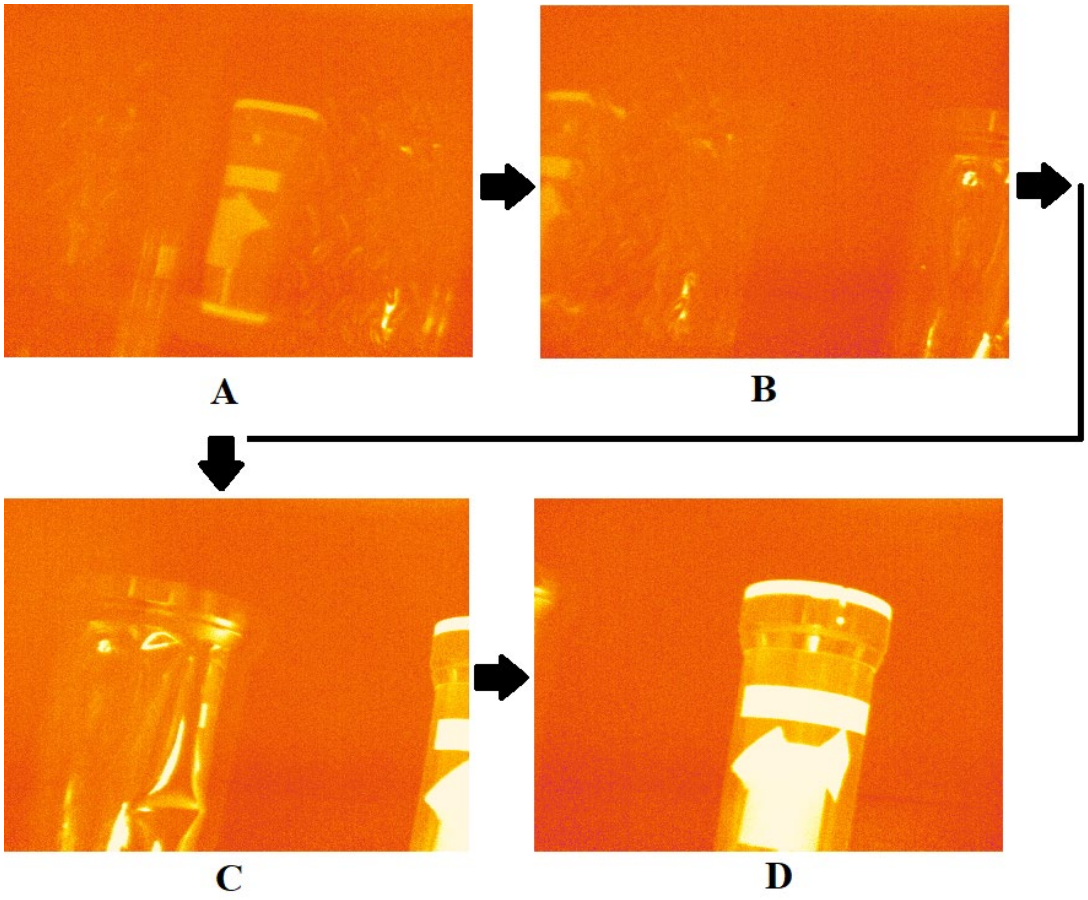


Figure 115. Thermal camera images showing thermal reflection of Pu-238 container in Storage cabinet 2 left door as arm pans to the right from A) Storage cabinet 2 left door (with the reflection), to B) Storage cabinet 2 left door and dented container, to C) dented container and left side of Pu-238 container, and to D) the actual Pu-238 container.

4.7 DISCUSSION

The integrated system test was successful in collecting neutron, visual, and thermal data from a realistic storage area in a radiological facility. Neutron count rates noticeably changed throughout the duration of the test, issues with damaged containers were able to be identified upon later operator review of the images, and the system did not collide with or otherwise damage its operating environment. However, more testing would be needed to assess what optimal performance would be in the future. The test environment was not as radiologically "crowded" with many individual sources as can be the case with storage locations at LANL. The current performance of the shield around the neutron detector does exhibit measurable differences in directionality, but these may be lost in the presence of many sources. Two actions should be taken to enhance the shielded neutron detector performance: improve the shielding and collect time longitudinal data for stored materials. The shielding could be augmented to increase the directionality by more effectively absorbing background neutrons. Increased shielding needs to continually be evaluated from a cost-benefit perspective to keep shielding size and weight manageable. The largest advantage comes from the collection of time longitudinal data where a baseline can be established. As repeated scans are collected over time, any changes to the environment are easier to detect. Changes to neutron counts would have to be evaluated to account for changing neutron background as material moves into or out of a storage area, but could also be a method to detect diversion or contamination (though the incorporation of alpha detectors with this system logically follows for searching for contamination). This is where the semiautonomous control schema is a boon because the robot can be designed to automatically flag radiological changes that should be reviewed by a human operator. The human operator can then determine whether an anomalous count rate is due to an approved material move or due to an unauthorized or unexpected move. The same concept of

collecting and evaluating data over time can be applied to the cameras and other sensors added in the future.

While not included in this work, repeating a scan of the integrated system in the test area without the shield around the detector would better elucidate the difference the directional shield provides. The shield is not perfectly directional, but experimental testing did confirm a measurable difference between neutrons incoming through the shield aperture versus other directions. Thus, it is anticipated that repeating the scan without the shield would produce an overall similarly-shaped curve that has been smoothed out when compared to Figure 99; reducing the distance between detector and source increases the count rate whether shielded or not, but the sharp differences between facing the source and rotating away would largely disappear, especially as seen at location D/storage cabinet 2 (not including the inherent detection efficiencies present in the geometry of the detector itself).

Chapter 5: Conclusions

5.1 Summary

Chapter 1 introduced the operating and regulatory environment present at LANL. It presented the motivation for increased monitoring of stored nuclear materials to improve the safety and reduce worker dose.

Chapter 2 reviewed previous work that has been done in relevant areas. It examined some of the previous attempts to measure the directionality of neutrons and attempts to control the direction of neutrons. It explored previous attempts to incorporate robotics in the nuclear field for surveillance, reconnaissance, and radiation detection and mapping. It also discussed some of the previously developed design principles and taxonomies for using layered levels of autonomy.

Chapter 3 presented the equipment and methodology used in this work, including the neutron detector, software, two robotic mobile platforms, robotic arm, and visual and thermal cameras. It also introduced the mathematical foundation behind creating an artificial neutron aperture for a detector and associated counting statistics.

Chapter 4 detailed the experimental results from this work (Table 7). The detector was carefully characterized, the shield simulated, and the assembly verified. The culmination was the integration of the neutron detector, visual camera, and thermal camera with a robotic mobile platform that successfully performed autonomous nuclear material surveys.

Table 7. Summary of Experiments Performed

Experiment	Equipment	Impact
Neutron detector spatial characterization	Benchtop neutron detector with PuBe source	Defined baseline spatial dependence of neutron detector
MCNP modeling	Computer simulation	Modeled performance of shielding around neutron detector; Results led to the addition of borated polyethylene layer to shielding
Shielded neutron detector spatial characterization	Benchtop neutron detector with fabricated shield and Cf-252 source	Measured the performance of the shield on detection directionality (~22% difference)
Pioneer motion planning	Pioneer LX platform	First test of semiautonomous control method with programming navigation between waypoints within a radiological facility
Integrated system test	Shielded neutron detector, UR5e arm, visual camera, and thermal camera mounted on Vector platform	Final test of all integrated components and their ability to perform a semiautonomous scan of radioactive materials within a radiological facility; System navigated between waypoints to collect neutron, visual, and thermal data on stored materials

5.2 FUTURE WORK

This work demonstrated the capability of a mobile robotic platform to be incorporated with a directional neutron detector to survey radiation. However, this was only the beginning of where this work could go. Below is a list of suggested ways to expand and improve upon what was presented here.

Detector reshape. It was desired to use a commercial off-the-shelf neutron detector for this work to explore the feasibility of the neutron shielding, but it is unlikely that the shape of the detector active volume is ideal for this application (for example, shortening the overall length could improve portability). In discussions with Bridgeport Instruments, they have confirmed that they could manufacture the same style of detector in a custom shape. More MCNP simulations would prove valuable to determine an optimal configuration of detector shape and shielding around it to maximize the directional detection for better source discrimination.

Detector movement. The detector was stationary as mounted on the Vector platform. It would be beneficial to be able to move the shielding aperture around. This would enable scanning in the up/Z-direction for materials stored at different levels. It could also be interesting to explore if the dip in detection efficiency along the long axis of the detection (as shown in Chapter 4) could be leveraged to increase the amount of directionality to detection when incorporated with the shielding. Additionally, rotational movement could potentially be incorporated with tomography techniques.

Arm movement. The motion planning for the UR5e in this work was relatively simple in that it moved between predetermined waypoints for each scan location in order to image the containers. This could be dramatically refined to include object recognition. Containers could be dynamically identified using visual feedback, and waypoints for inspection of identified containers could be calculated for the arm. Smarter arm movement

would increase autonomy and make scans more efficient by imaging containers rather than a whole storage location with any empty spaces.

Other radiation sensors. This work focused on the neutron detector, but other radiation sensors would eventually be integrated in a system designed to survey stored nuclear materials. To that effect, a gamma detector has been procured and is currently being investigated for incorporation with the Vector system.

Peripheral integration refinement. All the components in the integrated system test were mounted on the Vector platform in order to investigate their performance in a radiological environment. However, the design of this integration could be refined to provide an overall "cleaner" look. Also, the flashlight attached to the arm was useful in illuminating dark areas within the shelving, but a different type of light may be better. A ring light provides more diffused lighting that may help with specular reflectivity from metal containers.

Longitudinal data and machine learning. As part of the work with the Vector platform auxiliary to neutron detection, visual and thermal cameras were integrated with the system. It was always envisioned that the human review of images would be the initial proof of concept upon which computer analysis and machine learning could later be applied. Another aspect of visual, thermal, and radiological container inspection that could be expanded upon going forward is the value of time longitudinal data. Repeated scans offer a way to develop a characteristic baseline for a particular container. If changes start to appear over time, that may be a metric to incorporate to assess the overall container health or integrity. Similarly, as the detection and review methods for the camera data are developed and refined, it may be possible for a computer to detect container changes before they are visible to the human eye. The cameras have already begun to be used with machine learning to evaluate damage to storage containers. Similar techniques could be applied to

the radiation readings, or the radiation readings could be overlaid with the camera feedback.

Inventory information. Eventually it would be incredibly useful to be able to reference what items are stored where when mapping radiation information. This would enable the system to more easily determine if a large change in readings in one location is due to a problem or a result of items being moved. If inventory information were located on the robotic system, it would raise security and classification issues that would have to be addressed. However, inventory information could be kept separate from the robotic system and cross referenced with scan information *ex post facto* to circumvent security issues.

5.3 Concluding Remarks

This work presented an important first step in demonstrating the capability and potential of robotics for monitoring stored nuclear materials and increasing safety for human workers. Although there is a long history of incorporating robotics into the nuclear industry beginning in the 1980s, there is still a prevalent distrust of the reliability and usefulness of the systems. Past implementations have typically focused on collecting ambient radiation information (especially for gamma radiation). The novel contribution of this work lies in the incorporation of a directional neutron detector with a mobile robotic platform, and that it can also collect visual and thermal data. The security requirements at LANL also dictate a semiautonomous control scheme for radiation surveying with robotics. The classic choices of tether or wireless control were not appropriate to use in this work. The time-delay structure of issuing instructions and then requiring the robot to execute them autonomously will likely be the *de facto* standard for similar work until security requirements change. While it is the hope that one day wireless communications will be

allowed to expand the possibilities of robotics, this work has demonstrated that wireless technology is not strictly necessary to achieving improvements in the monitoring of stored nuclear materials.

Appendices

APPENDIX A

Appendix A contains sample code for an input deck for MCNP to model the neutron shielding.

```
Neutron Counter Cylindrical Shielding
c
c Poly Thickness = 2.54                    cm
c Cadmium Thickness = 0.0508                  cm
c Borated Poly Thickness = 2.54                     cm
c Window Angle = 1.047                   rad
c Detector Radius 2.413                   cm
c Detector Length = 35.56                   cm
c Poly Outer Radius = 4.953                   cm
c Cadmium Outer Radius = 5.0038                  cm
c Borated Poly Outer Radius = 7.5438
c Cut 1 X = 2.08948091303752
c Cut 1 Y = 1.20691280300273
c Cut 1 Z = 0
c Cut 2 X = -2.08948091303752
c Cut 2 Y = 1.20691280300273
c Cut 2 Z = 0
c Source X = 10.5438
c Source Y = 0
c Source Z = 0
c
c Cells
c
10 0             -10                     imp:n=0 $Inner Void
20 0              10 -20                 imp:n=1 $Inner Space
30 1 -0.97        20 -30                 imp:n=1 $Inner Poly
40 2 -8.65        30 -40                 imp:n=1 $Cadmium
c 41 3 -0.001225  30 -40 -51  52  53     imp:n=1 $Cadmium Gap
50 4 -1.00        40 -50 #51             imp:n=1 $Borated Poly
51 3 -0.001225    40 -50 -51  52  53     imp:n=1 $Borated Poly Gap
60 0             -60                     imp:n=0 $Lower Void End
Cap
61 0             -61                     imp:n=0 $Upper Void End
Cap
70 3 -0.001225    50 60 61 -70           imp:n=1 $Dry Air
80 0              70                     imp:n=0 $Void

c
```

```
c Surfaces
c
10 RCC 0 0 -17.78                0 0 35.56
2.4129
20 RCC 0 0 -17.78                0 0 35.56
2.413
30 RCC 0 0 -17.78                0 0 35.56
4.953
40 RCC 0 0 -17.78                0 0 35.56
5.0038
50 RCC 0 0 -17.78                0 0 35.56
7.5438
51 P 0 0 0 2.08948091303752
     1.20691280300273       0                      0 0 -1
52 P 0 0 0 -2.08948091303752
     1.20691280300273       0                      0 0 -1
53 PX 0
60 RCC 0 0 -17.791               0 0 0.01 7.5438
61 RCC 0 0 17.781                0 0 0.01 7.5438
70 SO 30.5438

c
c Data Cards
c
c Source Term
c
sdef pos = 10.5438               0 0 erg = d1
c sdef sur = 50 nrm = -1 erg= 2.5e-5 dir = d2
sp1 -3 1.025 2.926
c sb2  -21 2
c
c Materials
c
c HDPE 0.97 g/cc
m1     1001.80c  0.66667
       6000.80c  0.33333
mt1    poly.20t
c Cadmium 8.65 g/cc
m2     48106.80c 0.0125
       48108.80c 0.0089
         48110.80c 0.1249
         48111.80c 0.128
         48112.80c 0.2413
         48113.80c 0.1222
         48114.80c 0.2873
         48116.80c 0.0749
c Dry Air 0.001225 g/cc
c dry air (typical of American Southwest)
```

```
m3     1001.80c 1.7404E-10
       1002.80c 1.3065E-14
       2003.80c 8.3540E-16
       2004.80c 4.5549E-10
       6000.80c 1.11008E-08
       7014.80c 3.8981E-05
       7015.80c 1.3515E-07
       8016.80c 9.1205E-06
       8017.80c 3.4348E-09
       18036.80c 3.0439E-10
       18038.80c 5.3915E-11
       18040.80c 8.0974E-08
       36078.80c 1.7811E-14
       36080.80c 1.1164E-13
       36082.80c 5.6154E-13
       36083.80c 5.49985E-13
       36084.80c 2.69359E-12
       36086.80c 7.98498E-13
       54124.80c 2.30549E-13
mt3 lwtr.20t
c Borated Poly density 1.00 g/cc
m4     1001.80c 0.627759
       5010.80c 0.009338
        5011.80c 0.037352
        6000.80c 0.325552
c
c Tallies
c
f1:n 10.1
e1 0.001 21ilog 20
nps 10000000
```

APPENDIX B

Appendix B contains the xacro (XML macro) generated URDF (Unified Robot Description Format) file used in ROS for the Vector platform.

```
<?xml version="1.0" ?>
<!--
=================================================================
======================== -->
<!-- |    This document was autogenerated by xacro from
vector_ws/src/vector_common/vector_description/urdf/vector.
urdf.xacro | -->
<!-- |    EDITING THIS FILE BY HAND IS NOT RECOMMENDED
| -->
<!--
=================================================================
======================== -->
<!--
Software License Agreement (BSD)
\file      vector.urdf.xacro
\authors   DEVELOPER
\copyright Copyright (c) 2015, Stanley Innovation, Inc.,
All rights reserved.
Redistribution and use in source and binary forms, with or
without modification, are permitted provided that
the following conditions are met:
 * Redistributions of source code must retain the above
copyright notice, this list of conditions and the
   following disclaimer.
 * Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the
   following disclaimer in the documentation and/or other
materials provided with the distribution.
 * Neither the name of Stanley Innovation nor the names of
its contributors may be used to endorse or promote
   products derived from this software without specific
prior written permission.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WAR-
```

```
RANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, IN-
DIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
-->
<robot name="vector"
xmlns:xacro="http://ros.org/wiki/xacro">
  <!-- Included URDF/XACRO Files -->
  <gazebo>
    <plugin filename="libgazebo_ros_control.so"
name="gazebo_ros_control">
      <robotNamespace>/</robotNamespace>

<robotSimType>gazebo_ros_control/DefaultRobotHWSim</robotSi
mType>
      <legacyModeNS>true</legacyModeNS>
    </plugin>
    <plugin filename="libgazebo_ros_force_based_move.so"
name="object_controller">
      <commandTopic>/vector/cmd_vel</commandTopic>

<odometryTopic>/vector/odometry/local_filtered</odometryTop
ic>
      <odometryFrame>odom</odometryFrame>
      <odometryRate>100.0</odometryRate>
      <robotBaseFrame>base_link</robotBaseFrame>
      <publishOdometryTf>true</publishOdometryTf>
      <yaw_velocity_p_gain>8000.0</yaw_velocity_p_gain>
      <x_velocity_p_gain>10000.0</x_velocity_p_gain>
      <y_velocity_p_gain>10000.0</y_velocity_p_gain>
    </plugin>
  </gazebo>
  <!-- Base link is the center of the robot's chassis
between the motors projected on the ground -->
  <link name="base_link"/>
  <joint name="base_chassis_joint" type="fixed">
```

```
    <origin rpy="0 0 0" xyz="0 0 0.1143"/>
    <parent link="base_link"/>
    <child link="base_chassis_link"/>
  </joint>
  <link name="base_chassis_link">
    <inertial>
      <mass value="101.561195"/>
      <origin rpy="0 0 0" xyz="-0.000710 -0.000186
0.03813"/>
      <inertia ixx="3.126061" ixy="-0.032687" ixz="-
0.086662" iyx="-0.032687" iyy="13.356772" iyz="0.000468"
izx="-0.086662" izy="0.000468" izz="15.218700"/>
    </inertial>
    <collision>
      <geometry>
        <mesh
filename="package://vector_description/meshes/vector_compon
ents/collision/vector_base_collision.stl"/>
      </geometry>
    </collision>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh
filename="package://vector_description/meshes/vector_compon
ents/visual/vector_base.dae"/>
      </geometry>
    </visual>
  </link>
  <gazebo reference="base_chassis_link">
    <mu1 value="0.3"/>
    <mu2 value="0.3"/>
    <kp value="10000000.0"/>
    <kd value="1.0"/>
    <fdir1 value="1 0 0"/>
  </gazebo>
  <joint name="payload_plate_joint" type="fixed">
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <parent link="base_chassis_link"/>
    <child link="payload_plate_link"/>
  </joint>
  <link name="payload_plate_link">
    <inertial>
      <mass value="6.868901"/>
```

```
      <origin xyz="0.000219 0.000067 0.209254"/>
      <inertia ixx="0.115329" ixy="-0.000009"
ixz="0.000007" iyx="-0.000009" iyy="0.194172"
iyz="0.000002" izx="0.000007" izy="0.000002"
izz="0.309399"/>
    </inertial>
    <collision>
      <geometry>
        <mesh
filename="package://vector_description/meshes/vector_compon
ents/collision/std_payload_plate_collision.stl"/>
      </geometry>
    </collision>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh
filename="package://vector_description/meshes/vector_compon
ents/visual/std_payload_plate.dae"/>
      </geometry>
    </visual>
  </link>
  <gazebo reference="payload_plate_link">
    <mu1 value="0.3"/>
    <mu2 value="0.3"/>
    <kp value="10000000.0"/>
    <kd value="1.0"/>
    <fdir1 value="1 0 0"/>
  </gazebo>
  <link name="sic_imu_frame"/>
  <joint name="sic_imu_joint" type="fixed">
    <origin rpy="3.14159 0 1.570795" xyz="0.0326421
0.0899448 0.0082008"/>
    <parent link="base_chassis_link"/>
    <child link="sic_imu_frame"/>
  </joint>
  <gazebo>
    <plugin filename="libhector_gazebo_ros_imu.so"
name="sic_imu_controller">
      <alwaysOn>1</alwaysOn>
      <updateRate>50.0</updateRate>
      <bodyName>sic_imu_frame</bodyName>
      <topicName>/vector/feedback/sic_imu</topicName>
      <accelDrift>0.0005 0.0005 0.0005</accelDrift>
```

```
      <accelGaussianNoise>0.0005 0.0005
0.0005</accelGaussianNoise>
      <rateDrift>0.0005 0.0005 0.0005</rateDrift>
      <rateGaussianNoise>0.00005 0.00005 0.00005
</rateGaussianNoise>
      <headingDrift>0.00005</headingDrift>
      <headingGaussianNoise>0.00005</headingGaussianNoise>
    </plugin>
  </gazebo>
  <joint name="front_laser_joint" type="fixed">
    <origin rpy="3.14159 0 0.7853975" xyz="0.252995
0.192212 0.137128"/>
    <parent link="base_chassis_link"/>
    <child link="front_laser_link"/>
  </joint>
  <link name="front_laser_link">
    <visual>
      <geometry>
        <mesh
filename="package://vector_description/meshes/sensors/visua
l/hokuyo_uam_05lp.dae"/>
      </geometry>
    </visual>
    <collision>
      <geometry>
        <box size="0.05 0.05 0.05"/>
      </geometry>
    </collision>
  </link>
  <gazebo reference="front_laser_link">
    <sensor name="front_laser" type="ray">
      <pose>0 0 0 0 0 0</pose>
      <visualize>false</visualize>
      <update_rate>33.333</update_rate>
      <ray>
        <scan>
          <horizontal>
            <samples>720</samples>
            <resolution>1</resolution>
            <min_angle>-2.356</min_angle>
            <max_angle>2.356</max_angle>
          </horizontal>
        </scan>
        <range>
```

```
          <min>0.06</min>
          <max>20.0</max>
          <resolution>0.01</resolution>
        </range>
        <noise>
          <type>gaussian</type>
          <mean>0.0</mean>
          <stddev>0.01</stddev>
        </noise>
      </ray>
      <plugin filename="libgazebo_ros_laser.so"
name="front_laser_node">
        <topicName>/vector/front_scan</topicName>
        <frameName>front_laser_link</frameName>
      </plugin>
    </sensor>
  </gazebo>
  <joint name="rear_laser_joint" type="fixed">
    <origin rpy="3.14159 0 3.9269875" xyz="-0.252995 -
0.192212 0.137128"/>
    <parent link="base_chassis_link"/>
    <child link="rear_laser_link"/>
  </joint>
  <link name="rear_laser_link">
    <visual>
      <geometry>
        <mesh
filename="package://vector_description/meshes/sensors/visua
l/hokuyo_uam_05lp.dae"/>
      </geometry>
    </visual>
    <collision>
      <geometry>
        <box size="0.05 0.05 0.05"/>
      </geometry>
    </collision>
  </link>
  <gazebo reference="rear_laser_link">
    <sensor name="rear_laser" type="ray">
      <pose>0 0 0 0 0 0</pose>
      <visualize>false</visualize>
      <update_rate>33.333</update_rate>
      <ray>
        <scan>
```

```
          <horizontal>
            <samples>720</samples>
            <resolution>1</resolution>
            <min_angle>-2.356</min_angle>
            <max_angle>2.356</max_angle>
          </horizontal>
        </scan>
        <range>
          <min>0.06</min>
          <max>20.0</max>
          <resolution>0.01</resolution>
        </range>
        <noise>
          <type>gaussian</type>
          <mean>0.0</mean>
          <stddev>0.01</stddev>
        </noise>
      </ray>
      <plugin filename="libgazebo_ros_laser.so"
name="rear_laser_node">
        <topicName>/vector/rear_scan</topicName>
        <frameName>rear_laser_link</frameName>
      </plugin>
    </sensor>
  </gazebo>
  <joint name="vlp16_mount_joint" type="fixed">
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <parent link="base_chassis_link"/>
    <child link="vlp16_mount_link"/>
  </joint>
  <link name="vlp16_mount_link">
    <inertial>
      <mass value="12.0"/>
      <origin rpy="0 0 0" xyz="0 0 0.3635"/>
      <inertia ixx="0.335870" ixy="0.000000" ixz="0.000001"
iyx="0.000000" iyy="0.435105" iyz="0.000000" izx="0.000001"
izy="0.000000" izz="0.596237"/>
    </inertial>
    <collision>
      <geometry>
        <mesh
filename="package://vector_description/meshes/vector_compon
ents/collision/vector_vlp16_mount_collision.stl"/>
      </geometry>
```

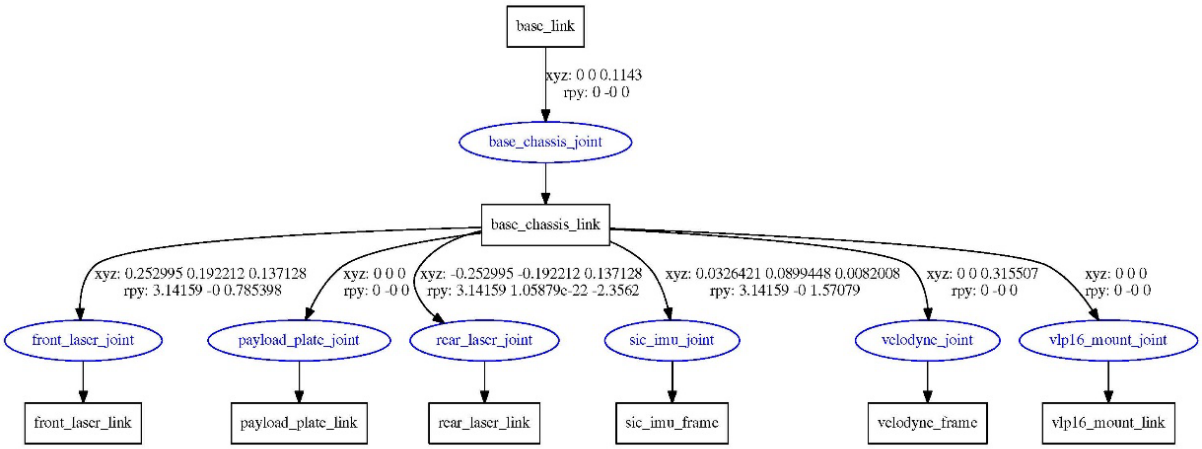
```
    </collision>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh
filename="package://vector_description/meshes/vector_compon
ents/visual/vector_vlp16_mount.dae"/>
      </geometry>
    </visual>
  </link>
  <gazebo reference="vlp16_mount_link">
    <mu1 value="0.3"/>
    <mu2 value="0.3"/>
    <kp value="10000000.0"/>
    <kd value="1.0"/>
    <fdir1 value="1 0 0"/>
  </gazebo>
  <joint name="velodyne_joint" type="fixed">
    <origin rpy="0 0 0" xyz="0 0 0.315507"/>
    <parent link="base_chassis_link"/>
    <child link="velodyne_frame"/>
  </joint>
  <link name="velodyne_frame">
    <inertial>
      <mass value="1.5"/>
      <origin xyz="0 0 0"/>
      <inertia ixx="0.001767" ixy="0.0" ixz="0.0" iyx="0.0"
iyy="0.001789" iyz="0.0" izx="0.0" izy="0.0"
izz="0.002159"/>
    </inertial>
    <visual>
      <geometry>
        <mesh
filename="package://vector_description/meshes/sensors/visua
l/velodyne_vlp16.dae"/>
      </geometry>
    </visual>
    <collision>
      <geometry>
        <mesh
filename="package://vector_description/meshes/sensors/colli
sion/velodyne_vlp16_collision.stl"/>
      </geometry>
    </collision>
```

```
  </link>
  <gazebo reference="velodyne_frame">
    &gt;

    <sensor name="velodyne" type="ray">
      <pose>0 0 0 0 0 0</pose>
      <visualize>false</visualize>
      <update_rate>10</update_rate>
      <ray>
        <scan>
          <horizontal>
            <samples>720</samples>
            <resolution>1</resolution>
            <min_angle>-3.14159</min_angle>
            <max_angle>3.14159</max_angle>
          </horizontal>
          <vertical>
            <samples>16</samples>
            <resolution>1</resolution>
            <min_angle>-0.261799388</min_angle>
            <max_angle>0.261799388</max_angle>
          </vertical>
        </scan>
        <range>
          <min>0.45</min>
          <max>150.0</max>
          <resolution>0.01</resolution>
        </range>
        <noise>
          <type>gaussian</type>
          <mean>0.0</mean>
          <stddev>0.01</stddev>
        </noise>
      </ray>
      <!-- test plugin -->
      <!-- <plugin name="plugin_1"
filename="libgazebo_ros_block_laser.so">
                    <updateRate>10.0</updateRate>

<topicName>/velodyne_pointcloud</topicName>
                    <frameName>/velodyne_frame</frameName>
                </plugin> -->
      <!-- sudo apt-get install ros-kinetic-velodyne-
gazebo-plugins -->
```

```
      <plugin filename="libgazebo_ros_velodyne_laser.so"
name="gazebo_ros_laser_controller">
        <topicName>/velodyne_points</topicName>
        <frameName>/velodyne_frame</frameName>
        <min_range>1</min_range>
        <max_range>80</max_range>
        <gaussianNoise>0.004</gaussianNoise>
      </plugin>
    </sensor>
  </gazebo>
</robot>
```

APPENDIX C

Appendix C contains the Graphviz diagram generated in ROS to represent the URDF file found in Appendix B.



APPENDIX D

Appendix D contains code that controls the operation of the neutron detector. This is a mix of Bridgeport Instruments included API and custom written code to poll the neutron detector during integrated system operation.

```
# release public
from __future__ import division
import ftdi
from ftdi import *
import time
import math
import os.path
from configobj import ConfigObj # to read .ini files


# module addresses
MA_CONTROLS   = 0; #!< Access eMorpho control registers
MA_STATISTICS = 1; #!< Access eMorpho statistics registers
MA_RESULTS    = 2; #!< Access eMorpho results registers (version,
telemtry, calibration)
MA_HISTOGRAM  = 3; #!< Access eMorpho histogram memory
MA_TRACE      = 4; #!< Access eMorpho trace memory
MA_LISTMODE   = 5; #!< Access eMorpho list mode memory
MA_USER       = 6; #!< Access eMorpho user memory
MA_MA7        = 7; #!< Access eMorpho firmware module no. 7 memory
(non-standard code)

def set_IS(IS, key, value):
      """ Use this function to avoid creating new keys in the IS
dictionary because of a typo."""
      if key in IS:
            IS[key] = value
      else:
            a = 1/0 # for debugging, raise an exception

def make_req(cmd_data, default_request):
      """Merge command data with comand defaults in case the input
data list is shorter than expected or supported.
      Also handles the case where the command data list is longer
than supported."""
      req = default_request #default record
      l_cmd = len(cmd_data)
      l_req = len(req)
      if(l_cmd)>= l_req:
            req = cmd_data[0:l_req]
```

```
      else:
            req[0:l_cmd] = cmd_data # replace the fields that were
given
      return req

def write(ft245, sn, module, words_in):
      nmax = len(words_in) // 16
      step = 16;
      words2ft245 = [0]*17
      words2ft245[0] = (module & 0xFF) * 0x100 + 1
      for n in range(nmax):
            if(n>0):
                  words2ft245[0] = (module & 0xFF) * 0x100 # no
address clear for subsequent writes
            for k in range(16):
                  words2ft245[k+1] = words_in[k + n*step]
            ft245.write_data(sn, words2ft245)

      return 0

def read(ft245, sn="", module=0, num_bytes=32, bytes_per_datum=2):
      PH = [(module & 0xFF) * 0x100 + 7] # short write, 2 bytes
packet header only
      ft245.write_data(sn, PH)   # program the module to read from
      return ft245.read_data(sn, num_bytes, bytes_per_datum)

def cr2is(CR, ADC_sampling_rate):
      """ Convert control register values into instrument settings.
This function defines the keys for IS"""
      if len(CR)<16:
            return dict()
      IS = dict() # We make IS a dictionary
      IS['fine_gain'] = int(CR[0])
      IS['baseline_threshold'] = int(CR[1]) & 0x03FF
      IS['pulse_threshold'] = int(CR[2]) & 0x03FF
      IS['hold_off'] = int(CR[3])
      IS['integration_time'] = int(CR[4])
      IS['roi_bounds'] = int(CR[5])
      IS['trigger_delay'] = int(CR[6]) & 0x03FF

      IS['dac_data'] = int(CR[7]) & 0xFFF;
      IS['HV'] = 3000.0/4096.0 * int(IS['dac_data'])

      IS['request'] = int(CR[8] + CR[9] * 0x10000)   #in units of
65536 adc_sampling clock cycles
      IS['ACQ_Time'] = 65536.0/ADC_sampling_rate * IS['request'];

      IS['pit'] = int(CR[10])
      IS['put'] = int(CR[11])
```

```
        # CR12
        val = int(CR[12])
        IS['ecomp'] = val & 0xF
        IS['pcomp'] = (val & 0xF0) >> 4
        IS['gain_select'] = (val & 0xF00) >> 8
        IS['lm_data_switch'] = (val & 0x1000) >> 12

        # CR13
        val = int(CR[13])
        IS['etout'] = val & 1
        IS['ptout'] = int((val & 0x2) / 0x2)
        IS['suspend'] = int((val & 0x4) / 0x4)
        IS['segment'] = int((val & 0x8) / 0x8)
        IS['segment_enable'] = int((val & 0x10) / 0x10)
        IS['daq_mode'] = int((val & 0x20) / 0x20)
        IS['nai_mode'] = int((val & 0x40) / 0x40)
        IS['temperature_disable'] = int((val & 0x80) / 0x80)

        # CR14
        val = int(CR[14])
        IS['opto_repeat_time'] = val & 0x1F
        IS['opto_pulse_width'] = (val >> 5) & 0xF
        IS['opto_pulse_sep'] = (val >> 9) & 0xF
        IS['opto_trigger'] = (val >> 14) & 0x1
        IS['opto_enable'] = (val >> 15) & 0x1

        # CR15
        val = int(CR[15])
        IS['clear_histogram']= val & 1
        IS['clear_statistics'] = int((val & 0x2)/0x2)
        IS['clear_trace'] = int((val & 0x4)/0x4)
        IS['clear_list_mode'] = int((val & 0x8)/0x8)
        IS['program_hv'] = int((val & 0x10)/0x10)
        IS['ut_run'] = int((val & 0x20)/0x20)
        IS['write_nv'] = int((val & 0x40)/0x40)
        IS['read_nv'] = int((val & 0x80)/0x80)

        IS['ha_run'] = int((val & 0x200)/0x200)
        IS['vt_run'] = int((val & 0x400)/0x400)
        IS['trace_run'] = int((val & 0x800)/0x800)
        IS['lm_run'] = int((val & 0x1000)/0x1000)
        IS['rtlt'] = int((val & 0x6000)/0x2000)
        IS['run'] = int((val & 0x8000)/0x8000)

        return IS

def is2cr(IS, ADC_sampling_rate):

        CR = [0]*16
        CR[0] = int(IS['fine_gain'])
```

```
      CR[1] = int(IS['baseline_threshold']) & 0x03FF
      CR[2] = int(IS['pulse_threshold']) & 0x03FF
      CR[3] = int(IS['hold_off'])
      CR[4] = int(IS['integration_time'])
      CR[5] = int(IS['roi_bounds'])
      CR[6] = int(IS['trigger_delay']) & 0x03FF

      dac_data = int(math.floor((4096.0/3000.0 * IS['HV'])+0.5)) &
0x0FFF
      CR[7] = dac_data;

      IS['request'] = int(math.floor(IS['ACQ_Time'] *
ADC_sampling_rate / 65536.0))
      CR[8] = int(IS['request']) & 0xFFFF                # lower 16-
bit word
      CR[9] = (int(IS['request'] & 0xFFFFFFFF)) >> 16    # upper 16-
bit word

      CR[10] = int(IS['pit'])
      CR[11] = int(IS['put'])

      val =  (int(IS['ecomp']) & 0xF) + ((int(IS['pcomp']) & 0xF) <<
4) + ((int(IS['gain_select']) & 0xF) << 8)
      val += (int(IS['lm_data_switch']) & 0x1) << 12
      CR[12] = val

      val =  (int(IS['etout']) & 1) + (int(IS['ptout']) & 1) * 0x2 +
(int(IS['suspend']) & 1) * 0x4 + (int(IS['segment']) & 1) * 0x8
      val += (int(IS['segment_enable']) & 1) * 0x10 +
(int(IS['daq_mode']) & 1) * 0x20 + (int(IS['nai_mode']) & 1) * 0x40
      val += (int(IS['temperature_disable']) & 1) * 0x80
      CR[13] = val

      val  = (int(IS['opto_repeat_time']) & 0x1F) +
((int(IS['opto_pulse_width']) & 0xF) << 5)
      val += (int(IS['opto_pulse_sep']) & 0xF) << 9
      val += ((int(IS['opto_trigger']) & 1) << 14) +
((int(IS['opto_enable']) & 1) << 15)
      CR[14] = val

      # self-clearing bits first
      val =  (int(IS['clear_histogram']) & 1) +
(int(IS['clear_statistics']) & 1) * 0x2 + (int(IS['clear_trace']) &
1) * 0x4
      val += (int(IS['clear_list_mode']) & 1) * 0x8 +
(int(IS['program_hv']) & 1) * 0x10 + (int(IS['ut_run']) & 1) * 0x20
      val += (int(IS['write_nv']) & 1) * 0x40 + (int(IS['read_nv'])
& 1) * 0x80
      # sticky bits
```

```
      val += (int(IS['ha_run']) & 1) * 0x200 + (int(IS['vt_run']) &
1) * 0x400 + (int(IS['trace_run']) & 1) * 0x800
      val += (int(IS['lm_run']) & 1) * 0x1000 + (int(IS['rtlt']) &
3) * 0x2000 + (int(IS['run']) & 1) * 0x8000
      CR[15] = val

      return CR; #!< convert instrument settings to control
registers;

def apply_settings(ft245, sn, IS, ss_time=0):
      """ ft245 contains the usb information for every attached
emorpho,
      sn is the serial number, IS are the instrument settings,
      and ss_time is the soft-start time."""

      IS['program_hv'] = 1;  # always update the high-voltage dac
      if ss_time == 0:
            CR = is2cr(IS, ft245.adc_speed[ft245.sn_to_devnum(sn)]);
# convert settings to control registers
            write(ft245, sn, MA_CONTROLS, CR)
      else:
            ss_time /= 10.0
            hv_max = IS['HV']
            hv_step = (hv_max // 10)
            hv_min = hv_max - 10*hv_step
            #for hv in range(hv_min,hv_max+1,hv-step):
            for n in range(1,11,1):
                  IS['HV'] = hv_min + n * hv_step
                  CR = is2cr(IS,
ft245.adc_speed[ft245.sn_to_devnum(sn)])   # convert settings to
control registers
                  write(ft245, sn, MA_CONTROLS, CR)
                  time.sleep(ss_time)
      # clear those bits that self-clear in the firmware
      IS['clear_histogram'] = 0
      IS['clear_statistics'] = 0
      IS['clear_trace'] = 0
      IS['clear_list_mode'] = 0
      IS['program_hv'] = 0
      IS['ut_run'] = 0
      IS['write_nv'] = 0
      IS['read_nv'] = 0
      return 0;


# The IS dictionary holds the instrument settings
# for data transfer that is numerical only, we need to convert
between the IS dictionary
# and the IS_val list.  A dictionary is not ordered, so val = [IS[k]
for k in IS] won't work
```

```
def IS_to_IS_val(IS):
      """Create a list from the values of IS in the order described
in the documentation."""
      IS_val =  [IS['fine_gain'], IS['baseline_threshold'],
IS['pulse_threshold']]
      IS_val += [IS['hold_off'], IS['integration_time'],
IS['roi_bounds'], IS['trigger_delay']]
      IS_val += [IS['dac_data'], IS['pit'], IS['put'],
IS['request'], IS['ecomp'], IS['pcomp']]
      IS_val += [IS['gain_select'], IS['lm_data_switch'],
IS['etout'], IS['ptout'], IS['suspend']]
      IS_val += [IS['segment'], IS['segment_enable'],
IS['daq_mode'], IS['nai_mode'], IS['temperature_disable']]
      IS_val += [IS['opto_repeat_time'], IS['opto_pulse_width'],
IS['opto_pulse_sep'], IS['opto_trigger'] ]
      IS_val += [IS['opto_enable'], IS['clear_statistics'],
IS['clear_histogram'], IS['clear_list_mode']]
      IS_val += [IS['clear_trace'], IS['ut_run'], IS['program_hv'],
IS['read_nv'], IS['write_nv'] ]
      IS_val += [IS['ha_run'], IS['trace_run'], IS['vt_run'],
IS['lm_run'], IS['rtlt'], IS['run'] ]
      IS_val += [IS['ACQ_Time'], IS['HV'] ]
      return IS_val

def IS_val_to_IS(IS_val):
      """Create an IS dictionary from the IS_val list."""
      IS = dict()
      IS['fine_gain'] = IS_val[0];      IS['baseline_threshold'] =
IS_val[1];
      IS['pulse_threshold'] = IS_val[2];IS['hold_off'] = IS_val[3];
      IS['integration_time'] = IS_val[4];     IS['roi_bounds'] =
IS_val[5];
      IS['trigger_delay'] = IS_val[6];  IS['dac_data'] = IS_val[7];

      IS['pit'] = IS_val[8]; IS['put'] = IS_val[9]; IS['request'] =
IS_val[10];

      IS['ecomp'] = IS_val[11];         IS['pcomp'] = IS_val[12];
      IS['gain_select'] = IS_val[13];   IS['lm_data_switch'] =
IS_val[14];

      IS['etout'] = IS_val[15];         IS['ptout'] = IS_val[16];
      IS['suspend'] = IS_val[17];
      IS['segment'] = IS_val[18];       IS['segment_enable'] =
IS_val[19];
      IS['daq_mode'] = IS_val[20]; IS['nai_mode'] = IS_val[21];
IS['temperature_disable'] = IS_val[22];

      IS['opto_repeat_time'] = IS_val[23];    IS['opto_pulse_width']
= IS_val[24];
```

```
      IS['opto_pulse_sep'] = IS_val[25];      IS['opto_trigger'] =
IS_val[26];
      IS['opto_enable'] = IS_val[27];

      IS['clear_statistics'] = IS_val[28];
      IS['clear_histogram'] = IS_val[29];
      IS['clear_list_mode'] = IS_val[30];
      IS['clear_trace'] = IS_val[31];              IS['ut_run'] =
IS_val[32];
      IS['program_hv'] = IS_val[33];               IS['read_nv'] =
IS_val[34]; IS['write_nv'] = IS_val[35];
      IS['ha_run'] = IS_val[36];        IS['trace_run'] =
IS_val[37];            IS['vt_run'] = IS_val[38];
      IS['lm_run'] = IS_val[39];        IS['rtlt'] = IS_val[40];
            IS['run'] = IS_val[41];
      IS['ACQ_Time'] = IS_val[42]; IS['HV'] = IS_val[43];
      return IS

# Read data from the eMorpho and put them in a form suitable to be
sent back to the client

# get and set for various settings groups

def get_sr(ft245,sn):
      """ Returns the ADC sampling rate in samples per second (Hz);
Not tied to a command """
      return ft245.adc_speed[ft245.sn_to_devnum(sn)]

def get_IS(ft245, sn):
      """ Returns the IS dictionary. It is not currently tied to a
command,
      but used often in the functions below. """
      CR = read(ft245, sn, MA_CONTROLS, 32, 2)
      IS = cr2is(CR, get_sr(ft245,sn))
      return IS

def boot_all(ft245, settings_path):
      for sn in ft245.sn:
            boot(ft245, sn, settings_path)
      return 0

def boot(ft245, sn, settings_path):
      # step 1: boot from NV-Mem
      download_nvmem(ft245, sn)
      # step 2: read sn-specific settings file, if it exists
      settings_file = settings_path + sn + '.ini'
      if(os.path.isfile(settings_file)): # If the file exists, merge
the instrument settings
            #print("found the individual settings file")
            config = ConfigObj(settings_file)
```

```
            is_config = config['Instrument_Settings'] # Section name
is Instrument_Settings
            IS = get_IS(ft245, sn)
            for key in is_config: # overwrite IS settings with those
read from file
                  if key in IS: # only accept keys that are already
in the IS dictionary
                        IS[key] = float(is_config[key])

      # step 3: read a settings file that applies to all units
      settings_file = settings_path + 'all_sn.ini'
      if(os.path.isfile(settings_file)): # If the file exists, merge
the instrument settings
            #print("found the global settings file")
            config = ConfigObj(settings_file)
            is_config = config['Instrument_Settings'] # Section name
is Instrument_Settings
            IS = get_IS(ft245, sn)
            for key in is_config: # overwrite IS settings with those
read from file
                  if key in IS: # only accept keys that are already
in the IS dictionary
                        IS[key] = float(is_config[key])
      # step 4: Apply seetings
      IS = get_IS(ft245, sn)
      apply_settings(ft245, sn, IS, ss_time=1.0) #ramp HV over 1
second
      get_status(ft245, sn) # to load the ADC sampling rate
      return 0

def get_is(ft245, sn):
      """ Returns a list of IS values in the order shown in the MDS
reference. """
      IS = get_IS(ft245, sn)  # retrieve the current IS
      IS_val = IS_to_IS_val(IS)
      return IS_val

def set_is(ft245, sn, cmd_data):
      """Receives an IS-values list, applies it to the eMorpho and
returns a list of 44 values."""
      IS_val = make_req(cmd_data, get_is(ft245, sn)) # Merge current
IS_val with given IS_val
      IS = IS_val_to_IS(IS_val)          # create new IS dictionary
      apply_settings(ft245, sn, IS)      # apply to emorpho
      IS_val = get_is(ft245, sn)         # read back to check
      return IS_val

def get_gain(ft245, sn):
      sr = get_sr(ft245,sn) # ADC sampling rate
      IS = get_IS(ft245, sn)  # retrieve the current IS
```

```
      dg = IS['fine_gain'] * pow(2.0, -IS['ecomp']) * sr/40.0e6
      gain = [IS['HV'], IS['gain_select'], dg ]
      return gain

def set_gain(ft245, sn, cmd_data):
      gain = make_req(cmd_data, get_gain(ft245, sn))
      sr = get_sr(ft245, sn) # ADC sampling rate
      IS = get_IS(ft245, sn)
      set_IS(IS,'HV',gain[0])
      set_IS(IS,'gain_select',gain[1])
      dg = gain[2]
      fg = dg * 40.0e6/sr;
      ecomp = 0
      while fg<16384:
            fg *= 2; ecomp += 1
            #print fg, ecomp

      set_IS(IS,'fine_gain',fg);
      set_IS(IS,'ecomp',ecomp);    # set_IS(IS,'pcomp',ecomp);
      CR = is2cr(IS, sr)
      apply_settings(ft245, sn, IS)
      gain = get_gain(ft245, sn)
      return gain

def get_dsp(ft245, sn):
      sr = get_sr(ft245,sn) # ADC sampling rate
      IS = get_IS(ft245, sn)
      put = IS['put']
      if(IS['nai_mode']==0):
            put = IS['put'] * 1.0e6/sr # convert to micro seconds

      roi_low  = (IS['roi_bounds'] & 0xFF) * 16;
      roi_high = (IS['roi_bounds'] &0xFF00)//0x100 * 16
      dsp = [IS['pulse_threshold'], IS['integration_time'] *
1.0e6/sr, IS['hold_off'] * 1.0e6/sr,\
               put, IS['baseline_threshold'], IS['pit'] * 1.0e6/sr,
IS['nai_mode'],\
               roi_low, roi_high, IS['temperature_disable'],
IS['suspend'] ]
      return dsp

def set_dsp(ft245, sn, cmd_data):
      dsp = make_req(cmd_data, get_dsp(ft245, sn))
      sr = get_sr(ft245, sn) # ADC sampling rate
      IS = get_IS(ft245, sn) # retrieve current IS dictionary
(instrument settings)

      # convert dsp to IS
      set_IS(IS, 'pulse_threshold', dsp[0])
      set_IS(IS, 'integration_time', round(dsp[1] * sr/1.0e6))
```

```
      set_IS(IS, 'hold_off', round(dsp[2] * sr/1.0e6))

      set_IS(IS, 'nai_mode', dsp[6])
      set_IS(IS, 'put', dsp[3])
      if IS['nai_mode']==0:
            set_IS(IS, 'put', round(dsp[3] * sr/1.0e6))
      set_IS(IS,'baseline_threshold', dsp[4])
      set_IS(IS, 'pit', round(dsp[5] * sr/1.0e6))
      set_IS(IS, 'roi_bounds', dsp[7]//16 + 0x100*(dsp[8]//16))
      set_IS(IS, 'temperature_disable', dsp[9])
      set_IS(IS, 'suspend', dsp[10])

      CR = is2cr(IS, sr)
      apply_settings(ft245, sn, IS)
      dsp = get_dsp(ft245, sn)
      return dsp

def get_pulser(ft245, sn):
      """ Use the firmware pulser to drive an LED."""
      IS = get_IS(ft245, sn)
      pulser = [IS['opto_repeat_time'], IS['opto_pulse_width'],
      IS['opto_pulse_sep'],\
                  IS['opto_trigger'], IS['opto_enable'] ]
      return pulser

def set_pulser(ft245, sn, cmd_data):
      """Program the paramaters controlling the firmware pulser"""
      pulser = make_req(cmd_data, get_pulser(ft245, sn))
      IS = get_IS(ft245, sn) # retrieve current IS dictionary
(instrument settings)
      sr = get_sr(ft245, sn)
      set_IS(IS, 'opto_repeat_time', pulser[0]);
      set_IS(IS, 'opto_pulse_width', pulser[1])
      set_IS(IS, 'opto_pulse_sep', pulser[2]);
      set_IS(IS, 'opto_trigger', pulser[3])
      set_IS(IS, 'opto_enable', pulser[4])
      CR = is2cr(IS, sr)
      apply_settings(ft245, sn, IS)
      pulser = get_pulser(ft245, sn)
      return pulser

def get_autocal(ft245, sn):
      return ft245.autocal[ft245.sn_to_devnum(sn)]

def set_autocal(ft245, sn, data):
      if(len(data)>6):
            data = data[0:6]
      autocal = get_autocal(ft245, sn)
      autocal[0:len(data)] = data[:]
      ft245.autocal[ft245.sn_to_devnum(sn)] = autocal[:]
```

```
def get_params(ft245, sn):
      """Return the four main parameter groups: gain, dsp, pulser
and autocal."""
      gain  = get_gain(ft245, sn)
      dsp     = get_dsp(ft245, sn)
      pulser  = get_pulser(ft245, sn)
      autocal = get_autocal(ft245, sn)
      return gain, dsp, pulser, autocal

def read_nvmem(ft245, sn):
      IS = get_IS(ft245, sn) # retrieve current IS dictionary
(instrument settings)
      set_IS(IS, 'read_nv', 1)
      apply_settings(ft245, sn, IS)
      time.sleep(0.02)
      nvmem = read(ft245, sn, MA_USER, 256, 2)
      return nvmem

def download_nvmem(ft245, sn):
      IS = get_IS(ft245, sn) # retrieve current IS dictionary
(instrument settings)
      set_IS(IS, 'read_nv', 1)
      apply_settings(ft245, sn, IS)
      time.sleep(0.02)
      nvmem = read(ft245, sn, MA_USER, 256, 2)
      if(nvmem[0] == 0x8003): #checks for valid nvmem content
(memory is not blank)
            CR = nvmem[1:17]
            write(ft245, sn, MA_CONTROLS, CR)
            autocal = nvmem[17:23]
            ft245.autocal[ft245.sn_to_devnum(sn)] = autocal[:]

      return nvmem

def update_nvmem(ft245, sn, data):
      CR = read(ft245, sn, MA_CONTROLS, 32, 2)
      autocal = ft245.autocal[ft245.sn_to_devnum(sn)]
      nvmem = [0]*128
      nvmem[0] = 0x8003
      nvmem[1:17] = CR[0:16]
      nvmem[17:23] = autocal[0:6]
      if(len(data)>64):
            data = data[0:64]
      nvmem[64:64+len(data)] =  data[:]
      nvmem = [int(d) for d in nvmem] # convert all to integer

      write(ft245, sn, MA_USER, nvmem)
      IS = get_IS(ft245, sn) # retrieve current IS dictionary
(instrument settings)
```

```
      set_IS(IS, 'write_nv', 1)
      apply_settings(ft245, sn, IS)
      time.sleep(0.02)
      return nvmem

def string_to_nvmem(ft245, sn, data_string):
      CR = read(ft245, sn, MA_CONTROLS, 32, 2)
      autocal = ft245.autocal[ft245.sn_to_devnum(sn)]
      nvmem = [0]*128
      nvmem[0] = 0x8003
      nvmem[1:17] = CR[0:16]
      nvmem[17:23] = autocal[0:6]
      if(len(data_string)>128):
            data_string = data_string[0:128]
      data_bytes = [ord(c) for c in data_string]
      if len(data_bytes)%2 == 1:
            data_bytes += [32]  # add a padding blank

      n_max = len(data_bytes)//2
      data = []
      for n in range(n_max):
            data += [data_bytes[2*n] + 256*data_bytes[2*n+1]]

      nvmem[64:64+len(data)] =  data[:]
      nvmem = [int(d) for d in nvmem] # convert all to integer

      write(ft245, sn, MA_USER, nvmem)
      IS = get_IS(ft245, sn) # retrieve current IS dictionary
(instrument settings)
      set_IS(IS, 'write_nv', 1)
      apply_settings(ft245, sn, IS)
      time.sleep(0.02)
      print nvmem
      return nvmem

def string_from_nvmem(ft245, sn):
      IS = get_IS(ft245, sn) # retrieve current IS dictionary
(instrument settings)
      set_IS(IS, 'read_nv', 1)
      apply_settings(ft245, sn, IS)
      time.sleep(0.02)
      nvmem = read(ft245, sn, MA_USER, 256, 2)

      data_bytes = []
      for n in range(64,128,1):
            data_bytes += [ nvmem[n] & 0x00FF, (nvmem[n] &
0xFF00)//256]
      data_bytes = [ d if d>0 else 32 for d in data_bytes] # replace
null-bytes with blanks
      data_string = ''.join(map(chr,data_bytes))
```

```
      return data_string

def adjust_led_target(ft245, sn, data):
      IS = get_IS(ft245, sn) # retrieve current IS dictionary
(instrument settings)
      factor = data[1]
      put_0 = IS['put']

      IS['put'] = put_0 * factor
      #set_IS(IS, 'put', 1)
      if data[3]>0:
            apply_settings(ft245, sn, IS)
            download_nvmem(ft245,sn)
            IS['put'] = put_0
            apply_settings(ft245, sn, IS)
      if data[2]>0:
            IS['put'] = put_0 * factor
            apply_settings(ft245, sn, IS)

      data_bytes = []
      data_bytes += [1,IS['put']]
      data_string = ' '.join(map(str,data_bytes))

      return data_string

# Get Rates (2 banks of statistics counters + 4 external counters)
def get_rates(ft245, sn):
      sr = get_sr(ft245, sn) # ADC Sampling Rate
      raw_stats = read(ft245,sn,MA_STATISTICS,64,4)
      rates = [0]*52 # prepare output data
      if(raw_stats[0] == 0):
            rates[0:10] = [0]*10
      else:
            rates[0:4] = raw_stats[0:4]         # DAQ time clock
ticks, events, triggers, dead time clock ticks (32-bit)
            rates[4] = raw_stats[1]             # histogrammed
events, usually same as raw_stats[1]
            rates[5] = raw_stats[0] * 65536/sr  # acquisition time in
seconds
            rates[6] = raw_stats[1] / rates[5]  # rate of
histogrammed events
            rates[7] = raw_stats[2] / rates[5]  # rate of recognized
triggers
            rates[8] = rates[7] / (1.0 - raw_stats[3]/raw_stats[0]) #
estimated true pulse rate (taking the dead time into account)
            rates[9] = raw_stats[3]/raw_stats[0] # acquisition dead
time fraction
```

```
      # Statistics associated with the second bank, when histogram
bank splitting is enabled
      # reporting 0 otherwise
      if raw_stats[4] == 0:
            rates[10:20] = [0]*10
      else:
            rates[10:14] = raw_stats[4:8]       # DAQ time clock
ticks, events, triggers, dead time clock ticks (32-bit)
            if(rates[10] == 0):
                  rates[14:20] = [0]*6
            else:
                  rates[14] = raw_stats[5]              #
histogrammed events, usually same as raw_stats[1]
                  rates[15] = raw_stats[4] * 65536/sr   # acquisition
time in seconds
                  rates[16] = raw_stats[5] / rates[15]  # rate of
histogrammed events
                  rates[17] = raw_stats[6] / rates[15]  # rate of
recognized triggers
                  rates[18] = rates[17] / (1.0 -
raw_stats[7]/raw_stats[4]) # estimated true pulse rate (taking the
dead time into account)
                  rates[19] = raw_stats[7] * 65536/sr   # acquisition
dead time in seconds

      # statistics for four additional counters (external digital
pulse source); bank 0
      for n in range(4):
            if rates[5] == 0:
                  rates[20+2*n] = 0    # Xctr_n events
                  rates[21+2*n] = 0    # Xctr_n rate (cps)
            else:
                  rates[20+2*n] = raw_stats[8+n]               #
Xctr_n events
                  rates[21+2*n] = raw_stats[8+n] / rates[5]    #
Xctr_n rate (cps)
      # statistics for four additional counters (external digital
pulse source); bank 1
      for n in range(4):
            if rates[15] == 0:
                  rates[36+2*n] = 0# Xctr_n events
                  rates[37+2*n] = 0# Xctr_n rate (cps)
            else:
                  rates[36+2*n] = raw_stats[12+n]              #
Xctr_n events
                  rates[37+2*n] = raw_stats[12+n] / rates[15]  #
Xctr_n rate (cps)
      return rates

# Get Rates (n banks of statistics counters )
```

```
def get_rates_b(ft245, sn, nb):
      sr = get_sr(ft245, sn) # ADC Sampling Rate
      raw_stats = read(ft245,sn,MA_STATISTICS,nb*16,4)
      rates = [0]*10*int(nb) # prepare output data

      for n in range(0,int(nb)):
            rates[0+10*n:4+10*n] = raw_stats[n*4:(n+1)*4]
            rates[4+10*n] = raw_stats[1+n*4]
            if raw_stats[4*n] != 0:
                  rates[5+10*n] = raw_stats[0+4*n] * 65536/sr  #
acquisition time in seconds
                  rates[6+10*n] = raw_stats[1+4*n] / rates[5+10*n]  #
rate of histogrammed events
                  rates[7+10*n] = raw_stats[2+4*n] / rates[5+10*n]  #
rate of recognized triggers
                  rates[8+10*n] = rates[7] / (1.0 -
raw_stats[3+4*n]/raw_stats[0+4*n]) # estimated true pulse rate
(taking the dead time into account)
                  rates[9+10*n] = raw_stats[3+4*n]/raw_stats[0+4*n] #
acquisition dead time fraction

      return rates

# Read version, status and calibration data and report in a format
to be sent to the client
def get_status(ft245, sn):
      raw_status = read(ft245,sn,MA_RESULTS,32)
      #print ', '.join(map(str, raw_status))
      CR = read(ft245,sn,MA_CONTROLS,32)
      sr = (raw_status[6] & 0xFF) * 1.0e6 # ADC sampling rate;
samples per second (Hz)
      if sr == 0: # USB read failed => device was unplugged
            status = [0]*22
            status[0] = 21
            return status
      ft245.adc_speed[ft245.sn_to_devnum(sn)] = sr  # Update the ADC
Sampling Rate (needed by cr2is, is2cr, etc)
      gain = (CR[12] // 0x100) & 0xF
      impedance = 100.0 + (gain & 1)*330.0 + (gain & 2)/2.0*1000.0
      impedance += (gain & 4)/4.0*3300.0 + (gain & 8)/8.0*10000.0
      max_ADC = 1024.0;  #  All on a 10-bit scale.  Extra bits are
fractional.
      ADC_voltage_range = 1.056;

      dc_val = raw_status[1] / 64     # DC-val; unit is 1.056mV
where full-scale is 1024*1.056mV.
      dI = ADC_voltage_range / max_ADC / impedance; # Anode current
per 10-bit ADC bin; in Ampere
      dQ = dI / sr      # Charge per 10-bit ADC bin and clock cycle;
in Coulomb
```

```
      d_mca_Q = dQ * (CR[12] & 0xF) / 2.0 * CR[0]/32768 # Charge per
MCA bin in Coulomb

      status = [0]*22 # prepare output data
      status[0]  = raw_status[7] & 0xFF;  #!< Firmware version
      status[1]  = raw_status[8]          #!< Customization number
      status[2]  = raw_status[9]          #!< Build number;
increases with every release
      status[3]  = (raw_status[7] // 256) & 0xF # number of ADC bits
      status[4]  = sr                     # ADC sampling rate;
samples per second (Hz)
      status[5]  = dc_val                 # DC-val; unit is 1.056mV
where full-scale is 1023*1.056mV.
      val        = raw_status[0]
      if val & 0x1000: # temperature is negative
            val = (val & 0x01FFF)-8192
      else:
            val = val & 0x07FF
      status[6]  = val / 16               # Temperature in deg. C
      status[7]  = raw_status[5]          # Average energy deposited
in the region of interest; reported as 16 times the average MCA bin.

      if(raw_status[4] & 0x1000):  # Test for sign bit; Negative
numbers mean zero current (just noise)
            status[8] = 0
      else:
            status[8] = (raw_status[3]+ 0x10000 * raw_status[4])* dI
* pow(2.0, -15.0)

      status[9] = (CR[15] & 0x8000) // 0x8000  # run active

      status[10] = raw_status[2] & 0x1        # histogram done
      status[11] = (raw_status[2] & 0x4) // 4 # trace done
      status[12] = (raw_status[2] & 0x2) // 2 # listmode done

      status[13] = impedance # Input amplifier transimpedance in
Ohms
      status[14] = max_ADC - dc_val
      status[15] = status[14] / 1000 * ADC_voltage_range / impedance
# Maximum measurable anode pulse current before going out of range
      status[16] = dQ # Charge unit: delta_T * delta_I
      status[17] = dI # Current unit: PMT-anode current per ADC bin
(ie per mV)
      status[18] = d_mca_Q # Charge unit per MCA bin: delta_q, in
Coulomb

      # Battery monitor
      status[19] = raw_status[10]
      status[20] = raw_status[11]
```

```
      # LED average
      status[21] = raw_status[12]
      return status

# Commands to perform data acquisition

def start_mca(ft245, sn, cmd_record):
      """Receives a start_mca record and restarts DAQ."""
      adc_sr = get_sr(ft245, sn)
      CR = read(ft245, sn, MA_CONTROLS, 32, 2)
      IS = cr2is(CR, adc_sr)
      req = make_req(cmd_record, [IS['ACQ_Time'], IS['rtlt'],
IS['ha_run'], 1, 1, IS['segment_enable'], 1]) # Merge command and
defaults as needed

      set_IS(IS, 'ACQ_Time', req[0])
      set_IS(IS, 'rtlt', req[1])
      set_IS(IS, 'ha_run', req[2])
      set_IS(IS, 'clear_histogram', req[3])
      set_IS(IS, 'clear_statistics', req[4])
      set_IS(IS, 'segment_enable', req[5])
      set_IS(IS, 'run', req[6])
      apply_settings(ft245, sn, IS)
      return 0

def start_trace(ft245, sn, cmd_data):
      adc_sr = get_sr(ft245, sn)
      CR = read(ft245, sn, MA_CONTROLS, 32, 2)
      IS = cr2is(CR, adc_sr)
      req = make_req(cmd_data, [0, IS['trigger_delay']]) # Merge
command and defaults as needed
      set_IS(IS, 'trigger_delay', req[1])
      if(req[0] == 0): # untriggered trace
            set_IS(IS, 'ut_run', 1)
            set_IS(IS, 'trace_run', 0)
            set_IS(IS, 'vt_run', 0)
      if(req[0] == 1): # triggered trace
            set_IS(IS, 'ut_run', 0)
            set_IS(IS, 'trace_run', 1)
            set_IS(IS, 'vt_run', 0)
      if(req[0] == 2): # validated and triggered trace
            set_IS(IS, 'ut_run', 0)
            set_IS(IS, 'trace_run', 0)
            set_IS(IS, 'vt_run', 1)
      set_IS(IS, 'clear_trace', 1)
      set_IS(IS, 'run', 1)
      apply_settings(ft245, sn, IS)

      return 0
```

```
def trace_summary(trace, IS, adc_sr):
      try:
            thr = IS['pulse_threshold']
            b_thr = IS['baseline_threshold']
            dc_val = trace[0]
            for n,t in enumerate(trace[1:]):
                  if abs(t-dc_val)<b_thr:
                        dc_val = 7/8*dc_val + t/8
                  elif (t-dc_val) > thr:
                        break
            else: # no pulse found
                  tlen = len(trace)
                  avg = sum(trace)/tlen
                  std_dev = math.sqrt(sum([(t-avg)**2/(tlen-1) for t
in trace]))
                  mini = min(trace)
                  maxi = max(trace)
                  return [adc_sr, -1, 0, 0, maxi, mini, std_dev, avg]

            delay = n+1 # trigger point
            tlen = len(trace)
            it = IS['integration_time']
            n0 = min(0,delay-8)
            n1 = max(min(0,delay-8+it),tlen-1)
            pulse = [t-dc_val for t in trace[n0:n1]]
            energy = sum(pulse)
            mca_bin = 32*(energy * IS['fine_gain']*pow(2.0, -
IS['ecomp'])) // (16*32768)

            ymax = max(pulse)
            xmax = pulse.index(ymax)

            y10 = 0.1*ymax
            y50 = 0.5*ymax
            y90 = 0.9*ymax

            p10 = [idx for idx,p in enumerate(pulse) if p>y10]
            xrise10 = p10[0]  + n0
            xfall10 = p10[-1] + n0

            p90 = [idx for idx,p in enumerate(pulse) if p>y90]
            xrise90 = p90[0]  + n0
            xfall90 = p90[-1] + n0

            rise_time = (xrise90 - xrise10)/adc_sr
            fall_time = (xfall10 - xfall90)/adc_sr
            peaking_time = (xmax - delay)/adc_sr

            p50 = [idx for idx,p in enumerate(pulse) if p>y50]
            fwhm = (p50[-1] - p50[0])/adc_sr
```

```
            results = [adc_sr, mca_bin, ymax, rise_time,
peaking_time, fall_time, fwhm, dc_val]
      except:
            results = [adc_sr, 0, 0, 0, 0, 0, 0, 0]
      #print ', '.join(map(str, results))

      return results

def start_lm(ft245, sn, cmd_data):
      """Receives command data list and starts a single listmode
run."""
      adc_sr = get_sr(ft245, sn)
      CR = read(ft245, sn, MA_CONTROLS, 32, 2)
      IS = cr2is(CR, adc_sr)
      req = make_req(cmd_data, [IS['lm_data_switch'],
IS['clear_statistics']]) # Merge command and defaults as needed
      set_IS(IS, 'lm_data_switch', req[0])
      set_IS(IS, 'lm_run', 1)
      set_IS(IS, 'clear_list_mode', 1)
      set_IS(IS, 'clear_statistics', req[1])
      set_IS(IS, 'run', 1)
      apply_settings(ft245, sn, IS)
      return 0

def exit_mds():
      sys.exit()
```

```
import os
import time
from  ctypes import *
import ctypes
import sys

#Interface between the API programmed in C and
#the python code.  Since complex objects, such
#as classes and structures don't travel well
#across such an interface, we only use variables
#and arrays.
#
#
#  We use ctypes to make the connection between
#python variables and C-variables.
LIBUSB_HAS_GET_DRIVER_NP = sys.platform.startswith('linux')

USE_WIN32 = sys.platform.startswith('win32') # rturns TRUE or FALSE

if sys.platform.startswith('linux'):
      w_dir=str(sys.path[0])
```

```
      print w_dir+"/lib/libusb.so"
      #Use the following two lines for linux
      cdll.LoadLibrary(w_dir+"/lib/libusb.so")
      libusb0=CDLL(w_dir+"/lib/libusb.so")
      _PATH_MAX=4096

elif sys.platform.startswith('win32'):
      #Use the following two line for windows
      w_dir=str(sys.path[0])
      #print w_dir+"\lib\libusb0_x86.dll"
      #cdll.LoadLibrary(w_dir+"\lib\libusb0_x86.dll")
      #libusb0=CDLL(w_dir+"\lib\libusb0_x86.dll")
      cdll.LoadLibrary("C:\Windows\System32\libusb0.dll")
      libusb0=CDLL("C:\Windows\System32\libusb0.dll")
      _PATH_MAX=511

# libusb-win32 makes all structures packed, while
# default libusb only does for some structures
# _PackPolicy defines the structure packing according
# to the platform.

class _PackPolicy(object):
      pass

if sys.platform == 'win32' or sys.platform == 'cygwin':
      _PackPolicy._pack_ = 1

class _usb_descriptor_header(Structure):
      _pack_ = 1
      _fields_ = [('blength', c_uint8),
                  ('bDescriptorType', c_uint8)]

class _usb_string_descriptor(Structure):
      _pack_ = 1
      _fields_ = [('bLength', c_uint8),
                  ('bDescriptorType', c_uint8),
                  ('wData', c_uint16)]

class _usb_endpoint_descriptor(Structure, _PackPolicy):
      _fields_ = [('bLength', c_uint8),
                  ('bDescriptorType', c_uint8),
                  ('bEndpointAddress', c_uint8),
                  ('bmAttributes', c_uint8),
                  ('wMaxPacketSize', c_uint16),
                  ('bInterval', c_uint8),
                  ('bRefresh', c_uint8),
                  ('bSynchAddress', c_uint8),
                  ('extra', POINTER(c_uint8)),
                  ('extralen', c_int)]
```

```
class _usb_interface_descriptor(Structure, _PackPolicy):
      _fields_ = [('bLength', c_uint8),
                        ('bDescriptorType', c_uint8),
                        ('bInterfaceNumber', c_uint8),
                        ('bAlternateSetting', c_uint8),
                        ('bNumEndpoints', c_uint8),
                        ('bInterfaceClass', c_uint8),
                        ('bInterfaceSubClass', c_uint8),
                        ('bInterfaceProtocol', c_uint8),
                        ('iInterface', c_uint8),
                        ('endpoint',
POINTER(_usb_endpoint_descriptor)),
                        ('extra', POINTER(c_uint8)),
                        ('extralen', c_int)]

class _usb_interface(Structure, _PackPolicy):
      _fields_ = [('altsetting',
POINTER(_usb_interface_descriptor)),
                        ('num_altsetting', c_int)]

class _usb_config_descriptor(Structure, _PackPolicy):
      _fields_ = [('bLength', c_uint8),
                        ('bDescriptorType', c_uint8),
                        ('wTotalLength', c_uint16),
                        ('bNumInterfaces', c_uint8),
                        ('bConfigurationValue', c_uint8),
                        ('iConfiguration', c_uint8),
                        ('bmAttributes', c_uint8),
                        ('bMaxPower', c_uint8),
                        ('interface', POINTER(_usb_interface)),
                        ('extra', POINTER(c_uint8)),
                        ('extralen', c_int)]

class _usb_device_descriptor(Structure, _PackPolicy):
      _pack_ = 1
      _fields_ = [('bLength', c_uint8),
                        ('bDescriptorType', c_uint8),
                        ('bcdUSB', c_uint16),
                        ('bDeviceClass', c_uint8),
                        ('bDeviceSubClass', c_uint8),
                        ('bDeviceProtocol', c_uint8),
                        ('bMaxPacketSize0', c_uint8),
                        ('idVendor', c_uint16),
                        ('idProduct', c_uint16),
                        ('bcdDevice', c_uint16),
                        ('iManufacturer', c_uint8),
                        ('iProduct', c_uint8),
                        ('iSerialNumber', c_uint8),
                        ('bNumConfigurations', c_uint8)]
```

```
class _usb_device(Structure, _PackPolicy):
      pass

class _usb_bus(Structure, _PackPolicy):
      pass

_usb_device._fields_ = [('next', POINTER(_usb_device)),
                                    ('prev', POINTER(_usb_device)),
                                    ('filename', c_int8 * (_PATH_MAX +
1)),
                                    ('bus', POINTER(_usb_bus)),
                                    ('descriptor',
_usb_device_descriptor),
                                    ('config',
POINTER(_usb_config_descriptor)),
                                    ('dev', c_void_p),
                                    ('devnum', c_uint8),
                                    ('num_children', c_ubyte),
                                    ('children',
POINTER(POINTER(_usb_device)))]

_usb_bus._fields_ = [('next', POINTER(_usb_bus)),
                              ('prev', POINTER(_usb_bus)),
                              ('dirname', c_char * (_PATH_MAX + 1)),
                              ('devices', POINTER(_usb_device)),
                              ('location', c_uint32),
                              ('root_dev', POINTER(_usb_device))]

_usb_dev_handle = c_void_p

libusb0.usb_get_busses.restype = ctypes.POINTER(_usb_bus)
libusb0.usb_get_busses.argtypes = []

libusb0.usb_open.restype = _usb_dev_handle
libusb0.usb_open.argtypes = [ctypes.POINTER(_usb_device)]

libusb0.usb_get_string_simple.argtypes = [_usb_dev_handle, c_int,
c_char_p, c_size_t ]

if(LIBUSB_HAS_GET_DRIVER_NP):
      libusb0.usb_detach_kernel_driver_np.argtypes =
[_usb_dev_handle, c_int]

libusb0.usb_set_configuration.argtypes = [_usb_dev_handle, c_int]
libusb0.usb_claim_interface.argtypes = [_usb_dev_handle, c_int]
libusb0.usb_control_msg.argtypes =\
      [_usb_dev_handle, c_int, c_int, c_int, c_int, c_char_p, c_int,
c_int]

libusb0.usb_close.argtypes = [_usb_dev_handle]
```

```
libusb0.usb_bulk_read.argtypes =\
      [_usb_dev_handle,c_int,c_char_p,c_int,c_int]

libusb0.usb_bulk_write.argtypes =\
      [_usb_dev_handle,c_int,c_char_p,c_int,c_int]

USB_RECIP_DEVICE = 0
USB_TYPE_VENDOR = 64
USB_ENDPOINT_OUT = 0
USB_ENDPOINT_IN = 0x80
FTDI_DEVICE_OUT_REQTYPE = (USB_TYPE_VENDOR | USB_RECIP_DEVICE |
USB_ENDPOINT_OUT)
FTDI_DEVICE_IN_REQTYPE = (USB_TYPE_VENDOR | USB_RECIP_DEVICE |
USB_ENDPOINT_IN)


#FT245 chip control I/O requests
SIO_SET_LATENCY_TIMER_REQUEST=0x9
SIO_GET_LATENCY_TIMER_REQUEST=0xA
SIO_READ_EEPROM_REQUEST=0x90
SIO_WRITE_EEPROM_REQUEST=0x91
SIO_ERASE_EEPROM_REQUEST=0x92

SIO_RESET_REQUEST=0x0

SIO_RESET_SIO=0x0
SIO_RESET_PURGE_RX=0x1
SIO_RESET_PURGE_TX=0x2

class ft245:
      def __init__(self):
            self.max_packet_size = 64
            self.in_ep  = 0x02  #!< writing to the FT245
            self.out_ep = 0x81  #!< reading from the FT245
            self.usb_read_timeout=1000
            self.usb_write_timeout=1000
            self.ftdi_vid=0x0403;
            self.bpi_vid=0x1FA4;
            self.morpho_pid=0x6001;
            self.use_only_bpi_vid = False;
            self.devices=list()
            self.handles=list()
            self.adc_speed=list()
            self.sn=list()

      def init(self):
            self.max_packet_size = 64
            self.in_ep  = 0x02  #!< writing to the FT245
            self.out_ep = 0x81  #!< reading from the FT245
```

```
            self.usb_read_timeout=1000
            self.usb_write_timeout=1000
            self.ftdi_vid=0x0403;
            self.bpi_vid=0x1FA4;
            self.morpho_pid=0x6001;
            self.use_only_bpi_vid = False;
            self.devices=list()
            self.handles=list()
            self.adc_speed=list()
            self.sn=list()

      def scan_all(self, bpi_only):
            """Scans for morphos and count them; Does not open a
Morpho."""
            libusb0.usb_init();
            libusb0.usb_find_busses()
            libusb0.usb_find_devices()
            bus = libusb0.usb_get_busses() # get won't work without
the find's above
            count = 0
            while(bool(bus)):
                  dev = bus[0].devices
                  while (bool(dev)):
                        vid = dev[0].descriptor.idVendor
                        pid = dev[0].descriptor.idProduct
                        ok = (vid == 0x1FA4 and pid == 0x6001) # find
emorpho with BPI VID
                        if( not bpi_only):
                              ok = ok or (vid == 0x0403 and pid ==
0x6001) # find emorpho with FTDI VID
                        if ok:
                              count +=1
                        dev = dev[0].next
                  bus = bus[0].next
            return count

      def find_all(self, bpi_only):
            libusb0.usb_init();
            libusb0.usb_find_busses()
            libusb0.usb_find_devices()
            bus = libusb0.usb_get_busses() # get won't work without
the find's above
            count = 0
            while(bool(bus)):
                  dev = bus[0].devices
                  while (bool(dev)):
                        vid = dev[0].descriptor.idVendor
                        pid = dev[0].descriptor.idProduct
                        ok = (vid == 0x1FA4 and pid == 0x6001) # find
emorpho with BPI VID
```

```
                        if( not bpi_only):
                              ok = ok or (vid == 0x0403 and pid ==
0x6001) # find emorpho with FTDI VID
                        if ok:
                              self.devices.append(dev)
                        dev = dev[0].next
                  bus = bus[0].next

            print "number of eMorphos: ",len(self.devices)
            if len(self.devices) == 0:
                  return -1

            self.handles = list()
            self.sn = list()
            dev = self.devices[0]
            offset = dev[0].descriptor.iSerialNumber
            for dev in self.devices:
                  handle = libusb0.usb_open(dev)
                  self.handles.append(handle) # open and store handle
                  sn_char = create_string_buffer('\000'*16)
                  libusb0.usb_get_string_simple(handle, offset,
sn_char, 16)
                  ser_num = ''.join(sn_char).split(b'\0',1)[0] #
treat first null-byte as stop character
                  self.sn.append(ser_num)

            # After the open, we need to detach the linux kernel
driver
            # and claim the interface
            for dev_num in range(len(self.devices)):
                  if(LIBUSB_HAS_GET_DRIVER_NP):

      libusb0.usb_detach_kernel_driver_np(self.handles[dev_num], 0)

                  if(USE_WIN32): # for ft245 config_value = 1 always

      #if(self.devices[dev_num][0].descriptor.bNumConfigurations >
0):
                        #     config_val =
self.devices[dev_num][0].config[0].bConfigurationValue
                        #     print "configuration value =
",config_val
                        #     if(config_val < 0):
                        #           return -1
                        #
      libusb0.usb_set_configuration(self.handles[dev_num],
config_val)

      libusb0.usb_set_configuration(self.handles[dev_num], 1)
```

```
      libusb0.usb_claim_interface(self.handles[dev_num],0)

                  # now reset the device
                  libusb0.usb_control_msg(
                        self.handles[dev_num], 64,0, 0, 0,
c_char_p(0), 0, self.usb_write_timeout)

                  latency = 2; # set 2ms minimum in all devices
                  self.set_latency_timer(latency)

            # now sort handles, serial numbers and other unit-
specific values
            num_devices = len(self.handles)
            sn_hndl = zip(self.sn, self.handles)
            sn_hndl_sorted = sorted(sn_hndl, key=lambda sn: sn[0]) #
sort opened emorphos by serial number
            self.sn2devnum = dict()
            for n in range(num_devices):
                  self.sn[n] = sn_hndl_sorted[n][0]
                  self.handles[n] = sn_hndl_sorted[n][1]
                  self.sn2devnum[self.sn[n]] = n  # Now build the S/N
to dev num dictionary
                  #print "S/N: ",self.sn[n]

      # preset default values for lists and values that are stored
within the ft245 class
            self.adc_speed = [40.0e6] * num_devices  # a default
value
            ac_list = [3.0, 1.0, 350, 1200, 661.62, 12]
            self.autocal = list()
            for n in range(num_devices):
                  self.autocal.append(ac_list)

            return 0

      # close all open devices
      def close(self):
            for handle in self.handles:
                  libusb0.usb_close(handle);
            self.handles = [] # now an empty list
            self.devices = [] # now an empty list
            return 0;

      # controls

      # set latency timer in all devices
      def set_latency_timer(self, latency):
            latency = latency & 0xFF; # only lower byte is valid
            for handle in self.handles:
```

```
                libusb0.usb_control_msg(handle, 64,

      SIO_SET_LATENCY_TIMER_REQUEST,
                                        latency, 0, c_char_p(0), 0,
self.usb_write_timeout);
            return 0;

      def sn_to_devnum(self, sn):
            """ Find the ordinal number of the unit.
            If sn is an empty string, we use devnum=0 as the default.
            If a serial number can't be found, we return devum=-1"""
            if len(sn)>0:
                  if sn in self.sn2devnum:
                        devnum = self.sn2devnum[sn]
                  else:
                        devnum = -1
            else:
                  devnum = 0
            return devnum

      # purge receive buffer in enumerated device
      def purge_rx_buffer(self, sn):
            dev_num = self.sn_to_devnum(sn)
            handle = self.handles[dev_num]
            ret = libusb0.usb_control_msg(handle, 64,
                                        SIO_RESET_REQUEST,
SIO_RESET_PURGE_RX,
                                        0, c_char_p(0), 0,
self.usb_write_timeout);
            return ret;

      # read byte data from device dev_num
      def read_data(self, sn, num_bytes, bytes_per_datum):
            dev_num = self.sn_to_devnum(sn)
            read_bytes = int(((num_bytes + 256)//62 + 1)*64)
            char_buf = create_string_buffer('\000'*read_bytes)
            handle = self.handles[dev_num]
            self.purge_rx_buffer(sn)
            ret = libusb0.usb_bulk_read (
                        handle, self.out_ep, char_buf, read_bytes,
self.usb_read_timeout);

            # Remove the modem status bytes
            bytes_out = []
            for n in range(read_bytes):
                  if( (n%64 == 0) or (n%64 == 1) ): continue;
                  bytes_out.append(char_buf[n])

            # Combine bytes into data words (16=bit or 32-bit)
            data_out = []
```

```
            if(bytes_per_datum == 2):
                  for n in range(int(num_bytes//2)): # '//' is the
floor division operator
                        data_out.append(ord(bytes_out[2*n+256]) +
0x100 * ord(bytes_out[2*n+257]))

            if(bytes_per_datum == 4):
                  for n in range(int(num_bytes//4)): # '//' is the
floor division operator
                        data_out.append(ord(bytes_out[4*n+256]) +
0x100 * ord(bytes_out[4*n+257]) +

      0x10000*ord(bytes_out[4*n+258]) + 0x1000000 *
ord(bytes_out[4*n+259]))

            return data_out;

      # write data to emorpho dev_num
      def write_data(self, sn, words_in):
            num_words = len(words_in)
            dev_num = self.sn_to_devnum(sn)
            handle = self.handles[dev_num]
            buf = create_string_buffer('\000'*(num_words*2-1))
            for n in range(num_words):
                  buf[2*n]   = chr(words_in[n] % 0x100);
                  buf[2*n+1] = chr(words_in[n] // 0x100);
            ret = libusb0.usb_bulk_write(handle, self.in_ep, buf,
num_words*2,

      self.usb_write_timeout);
            #wr_buf = [ord(buf[n]) for n in range(num_bytes)]
            #print "write buffer = ", wr_buf
            return ret;

      # read the eeprom of emorpho number dev_num
      def read_eeprom(self, sn):
            dev_num = self.sn_to_devnum(sn)
            buf = ctypes.create_string_buffer('\000'*256)
            ret = 0
            pval = ctypes.c_char_p(0)
            handle = self.handles[dev_num]
            eeprom = [0]*128
            #cprom = [' ']*128
            for n in range(64):
                  ret += libusb0.usb_control_msg( # can deliver only
2 bytes at a time
                        handle, FTDI_DEVICE_IN_REQTYPE,
                        SIO_READ_EEPROM_REQUEST, 0,
                        n, buf, 2,
                        self.usb_write_timeout)
```

```
                #print "{0:X},
{1:X}".format(*map(int,map(ord,buf[0:2])))
                eeprom[2*n:2*n+2] = map(int,map(ord,buf[0:2]))
                #cprom[2*n:2*n+2] = buf[0:2]
                #print cprom
          return eeprom

     # write eeprom for emorpho number dev_num
     def write_eeprom(self, sn, eeprom):
          dev_num = self.sn_to_devnum(sn)
          handle = self.handles[dev_num]
          self.set_latency_timer(0x77) # writing to the eeprom
takes time
          buf = ctypes.create_string_buffer('\000'*256)
          for n in range(64):
                usb_val = eeprom[2*n] + 0x100 * eeprom[2*n+1]; #
low-byte first
                ret = libusb0.usb_control_msg(
                      handle, FTDI_DEVICE_OUT_REQTYPE,
                      SIO_WRITE_EEPROM_REQUEST, usb_val, n,  # n
fills the index slot.
                      buf, 0, self.usb_write_timeout);

          self.set_latency_timer(2);
          return 0;


     def ftdi_eeprom_encode(self, usb_id):
          """Build binary output from usb_id dictionary. Output is
suitable for write_eeprom()
          returns an array of 128 byte-values"""

          # size check

          eeprom_size = 128
          esize = 28 + 2*(int(usb_id['mfg_size']) +
int(usb_id['prod_size']) + int(usb_id['sernum_size']))+6
          if(esize > eeprom_size):
                return False, []

          eeprom = [0]*128

          eeprom[0] = (int(usb_id['first']) & 0xFF)
          eeprom[1] = (int(usb_id['first'])&0xFF00)//0x100
          eeprom[2] = int(usb_id['vendor_id']) & 0xFF # Addr 02:
Vendor ID
          eeprom[3] = (int(usb_id['vendor_id']) & 0xFF00)//0x100
          eeprom[4] = int(usb_id['product_id']) & 0xFF # Addr 02:
Vendor ID
          eeprom[5] = (int(usb_id['product_id']) & 0xFF00)//0x100
```

```
            eeprom[6] = int(usb_id['chip_type']) & 0xFF  # Addr 06:
Device release number (0400h for BM features)
            eeprom[7] = (int(usb_id['chip_type']) & 0xFF00)//0x100
            # Addr 08: Config descriptor
            # Bit 7: always 1
            # Bit 6: 1 if this device is self powered, 0 if bus
powered
            # Bit 5: 1 if this device uses remote wakeup
            # Bit 4: 1 if this device is battery powered
            eeprom[8] = int(usb_id['power_config'])
            # Addr 09: Max power consumption: max power = value * 2
mA
            eeprom[9] = int(usb_id['max_current'])//2

            # Addr 0A: Chip configuration
            # Bit 7: 0 - reserved
            # Bit 6: 0 - reserved
            # Bit 5: 0 - reserved
            # Bit 4: 1 - Change USB version
            # Bit 3: 1 - Use the serial number string
            # Bit 2: 1 - Enable suspend pull downs for lower power
            # Bit 1: 1 - Out EndPoint is Isochronous
            # Bit 0: 1 - In EndPoint is Isochronous
            # Addr 0B: reserved
            eeprom[10] = int(usb_id['chip_config'])
            eeprom[11] = int(usb_id['addr_0xb'])


            # Addr 0C: USB version low byte when 0x0A bit 4 is set
            # Addr 0D: USB version high byte when 0x0A bit 4 is set
            eeprom[12] = int(usb_id['usb_version']) & 0xFF
            eeprom[13] = (int(usb_id['usb_version']) & 0xFF00)//0x100

            # 4 undocumented bytes at address 0x14 to 0x17
            eeprom[0x14] = int(usb_id['addr_0x14'])
            eeprom[0x15] = int(usb_id['addr_0x15'])
            eeprom[0x16] = int(usb_id['addr_0x16'])
            eeprom[0x17] = int(usb_id['addr_0x17'])

            # Each of the three string sections below is 2 bytes
longer than the string itself.
            # Addr 0E: Offset of the manufacturer string + 0x80
            # Addr 0F: Length of manufacturer string
            off = 24
            L = len(usb_id['mfg'])
            eeprom[14] = (off | 0x80)
            eeprom[15] = 2*L+2
            eeprom[off] = 2*L+2
            eeprom[off+1] = 0x03
            lst = map(ord,list(usb_id['mfg']))
```

```
            for n in range(L):
                  eeprom[off+2+2*n]   = lst[n]
                  eeprom[off+2+2*n+1] = 0

            # Addr 0x10: Offset of the product string + 0x80
            # Addr 0x11: Length of product string
            off += 2*L+2
            L = len(usb_id['product'])
            eeprom[16] = (off | 0x80)
            eeprom[17] = 2*L+2
            eeprom[off] = 2*L+2
            eeprom[off+1] = 0x03
            lst = map(ord,list(usb_id['product']))
            for n in range(L):
                  eeprom[off+2+2*n]   = lst[n]
                  eeprom[off+2+2*n+1] = 0

            # Addr 0x12: Offset of the serial string + 0x80
            # Addr 0x13: Length of serial string
            off += 2*L+2
            L = len(usb_id['sernum'])
            eeprom[18] = (off | 0x80)
            eeprom[19] = 2*L+2
            eeprom[off] = 2*L+2
            eeprom[off+1] = 0x03
            lst = map(ord,list(usb_id['sernum']))
            for n in range(L):
                  eeprom[off+2+2*n]   = lst[n]
                  eeprom[off+2+2*n+1] = 0

            # verify checksum; confine computations to 16-bit
            checksum = 0xAAAA;
            for i in range(63):
                  value = eeprom[2*i] + 0x100*eeprom[2*i+1]
                  checksum = value^checksum
                  checksum = ((checksum << 1)&0xFFFF) | (checksum >>
15)

            eeprom[126] = checksum & 0xFF
            eeprom[127] = (checksum & 0xFF00) >> 8

            return True, eeprom

      def ftdi_eeprom_decode(self, eeprom):
            """ Decode binary EEPROM image into an ftdi_eeprom
structure.
            eeprom is an array of 64 16-bit words;
            returns OK, usb_id: OK if checksum is OK, human-readable
usb_id dictionary
            """
```

```
            usb_id = {}

            # Addr 00: Stay 00 00
            usb_id['first'] = eeprom[0] + 0x100*eeprom[1]

            # Addr 02: Vendor ID
            usb_id['vendor_id'] = eeprom[2] + 0x100*eeprom[3]

            # Addr 04: Product ID
            usb_id['product_id'] = eeprom[4] + 0x100*eeprom[5]

            usb_id['chip_type'] = eeprom[6] + 0x100*eeprom[7]


            # Addr 08: Config descriptor
            # Bit 7: always 1
            # Bit 6: 1 if this device is self powered, 0 if bus
powered
            # Bit 5: 1 if this device uses remote wakeup
            # Bit 4: 1 if this device is battery powered
            usb_id['power_config'] = eeprom[8] & 0x00FF

            # Addr 09: Max power consumption: max power = value * 2
mA
            usb_id['max_current'] = 2*eeprom[9] # in mA


            # Addr 0A: Chip configuration
            # Bit 7: 0 - reserved
            # Bit 6: 0 - reserved
            # Bit 5: 0 - reserved
            # Bit 4: 1 - Change USB version
            # Bit 3: 1 - Use the serial number string
            # Bit 2: 1 - Enable suspend pull downs for lower power
            # Bit 1: 1 - Out EndPoint is Isochronous
            # Bit 0: 1 - In EndPoint is Isochronous
            # Addr 0B: reserved
            usb_id['chip_config'] = eeprom[10] # store lower and
upper byte
            usb_id['addr_0xb'] = eeprom[11]

            # Addr 0C: USB version low byte when 0x0A bit 4 is set
            # Addr 0D: USB version high byte when 0x0A bit 4 is set
            usb_id['usb_version'] = eeprom[12] + 0x100*eeprom[13]

            # Each of the three string sections below is 2 bytes
longer than the string itself.
            # Addr 0E: Offset of the manufacturer string + 0x80,
calculated later
```

```
            usb_id['mfg_off'] = eeprom[14] - 0x80
            # Addr 0F: Length of manufacturer string
            usb_id['mfg_size'] = (eeprom[15]-2)//2 # length of the
actual string. (it takes twice as many bytes in the eeprom.)
            # Addr 0x10: Offset of the product string + 0x80,
calculated later
            usb_id['prod_off'] = eeprom[16] - 0x80
            # Addr 0x11: Length of product string
            usb_id['prod_size'] = (eeprom[17]-2)//2

            # Addr 0x12: Offset of the serial string + 0x80,
calculated later
            usb_id['sernum_off'] = eeprom[18] - 0x80
            # Addr 0x13: Length of serial string
            usb_id['sernum_size'] = (eeprom[19]-2)//2

            # Decode manufacturer
            off = usb_id['mfg_off'] + 2
            lst = [eeprom[2*n+off] for n in
range(usb_id['mfg_size'])]
            usb_id['mfg'] = ''.join(map(chr,lst))

            # Decode product name
            off = usb_id['prod_off'] + 2
            lst = [eeprom[2*n+off] for n in
range(usb_id['prod_size'])]
            usb_id['product'] = ''.join(map(chr,lst))

            # Decode product name
            off = usb_id['sernum_off'] + 2
            lst = [eeprom[2*n+off] for n in
range(usb_id['sernum_size'])]
            usb_id['sernum'] = ''.join(map(chr,lst))

            # record undocumented 4 bytes
            usb_id['addr_0x14'] = eeprom[0x14]
            usb_id['addr_0x15'] = eeprom[0x15]
            usb_id['addr_0x16'] = eeprom[0x16]
            usb_id['addr_0x17'] = eeprom[0x17]

            # verify checksum; confine computations to 16-bit
            checksum = 0xAAAA;
            for i in range(63):
                  value = eeprom[2*i] + 0x100*eeprom[2*i+1]
                  checksum = value^checksum
                  checksum = ((checksum << 1)&0xFFFF) | (checksum >>
15)

            eeprom_checksum = eeprom[126] + 0x100*eeprom[127]
            OK = True if eeprom_checksum == checksum else False
```

```
            return OK, usb_id
```

```
#!/usr/bin/python
#
# version 1.0
from __future__ import division
import zmq
import time
import sys
import emorpho_io
import ftdi
import argparse

def rates_daq(dwell_time):
      print("rates dwell_time: "+ str(dwell_time))
      """ Boot eMorpho, acquire rates, save data and exit """
      use_bpi_vid_only = 1 # only recognize devices with te BPI
vendor id
      settings_path = 'settings/'
      all_morpho = ftdi.ft245()

      ret_find = all_morpho.find_all(use_bpi_vid_only)
      if(ret_find >= 0):
            emorpho_io.boot_all(all_morpho, settings_path)

      #print 'serial numbers = ',all_morpho.sn
      sn = all_morpho.sn[0] # use the first device only
      print ("active device: ", sn)

      time.sleep(5) # Wait for high voltage to ramp up.

      # start new MCA acquisition
      cmd_record = [1, 0, 0, 1, 1, 0, 1]
      emorpho_io.start_mca(all_morpho, sn, cmd_record)

      time.sleep(dwell_time) # Acquire a rates for dwell_time
seconds
      rates  = emorpho_io.get_rates_b(all_morpho, sn, 1)

      data = ','.join(map(str,rates))
      with open('{0}_rates.csv'.format(sn),'a') as fout:
            fout.write(data+'\n')

def multi_rates_daq(dwell_time, max_time):
      print("multi_rates dwell_time: " + str(dwell_time) +",
max_time: "+ str(max_time))
```

```
      """ Boot eMorpho, acquire rates, save data every dwell_time
seconds;
          stops when max_time is reached or exceeded; all times are
in seconds"""
      use_bpi_vid_only = 1 # only recognize devices with te BPI
vendor id
      settings_path = 'settings/'
      all_morpho = ftdi.ft245()

      ret_find = all_morpho.find_all(use_bpi_vid_only)
      if(ret_find >= 0):
            emorpho_io.boot_all(all_morpho, settings_path)

      #print 'serial numbers = ',all_morpho.sn
      sn = all_morpho.sn[0] # use the first device only
      print ("active device: ",sn)

      time.sleep(5) # Wait for high voltage to ramp up.

      # start new MCA acquisition
      cmd_record = [1, 0, 0, 1, 1, 0, 1]
      emorpho_io.start_mca(all_morpho, sn, cmd_record)
      then = time.clock()
      while True:
            time.sleep(dwell_time)
            rates  = emorpho_io.get_rates_b(all_morpho, sn, 1)

            data = ','.join(map(str,rates))
            with open('{0}_rates.csv'.format(sn),'a') as fout:
                  fout.write(data+'\n')
            #emorpho_io.start_mca(all_morpho, sn, cmd_record) #
restart if you don't want spectra to accumulate
            if time.clock() - then > max_time:
                  break

#rates_daq(10)
#multi_rates_daq(dwell_time=10, max_time=100)

parser = argparse.ArgumentParser()
parser.add_argument("--dwell_time", type=int, help="dwell_time
seconds", required=True)
parser.add_argument("--max_time", type=int, help="max_time seconds",
required=False)
args = parser.parse_args()
if not args.max_time:
      rates_daq(args.dwell_time)
else:
      multi_rates_daq(dwell_time=args.dwell_time,
max_time=args.max_time)
```

References

Adept MobileRobotics. (2013). *Pioneer LX User's Guide Rev A*.

Balmer, M. J. I., Gamage, K. A. A., & Taylor, G. C. (2014). Critical review of directional neutron survey meters. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *735*, 7–11. https://doi.org/10.1016/j.nima.2013.08.087

Brabec, C., Schroeder, K., Williams, J., O'Neil, B., Hashem, J., & Pryor, M. (2011). Reducing the operator's burden during teleoperation involving contact tasks. *Proc. of 3rd Int. Joint Topical Meeting on Emergency Preparedness and Response and Robotics and Remote Systems, EPRRSD, 13th Robotics and Remote Systems for Hazardous Environments and 11th Emergency Preparedness and Response,* 202-212.

Bridgeport Instruments, LLC. (2012, March 14). R2DNT Neutron Detector [Fact sheet]. Retrieved from http://www.bridgeportinstruments.com/products/ncounter/ndet_2x24_r1.pdf

Brown, W. G. (2004). *Performance Testing of a Robotic Lathe at Los Alamos National Laboratory (LA-UR-04-1617)*. LANL.

Bruemmer, D. J., Few, D. A., Boring, R. L., Marble, J. L., Walton, M. C., & Nielsen, C. W. (2005). Shared Understanding for Collaborative Control. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 35(4), 494–504. https://doi.org/10.1109/tsmca.2005.850599

Caglioti, G., & Casali, F. F. (1962). Rubber Soller Slit Collimators for Neutron Spectrometry. *Review of Scientific Instruments*, *33*(10), 1103–1105. https://doi.org/10.1063/1.1717693

Connette, C., Marthi, B., & Khandelwal, P. (2018). eband_local_planner - ROS Wiki. Retrieved from Ros.org website: http://wiki.ros.org/eband_local_planner

Cortez, R. A., Tanner, H. G., & Lumia, R. (2009). Distributed Robotic Radiation Mapping. Experimental Robotics, 147–156. https://doi.org/10.1007/978-3-642-00196-3_17

Currie, L. A. (1968). Limits for qualitative detection and quantitative determination. Application to radiochemistry. *Analytical Chemistry*, *40*(3), 586–593. https://doi.org/10.1021/ac60259a007

Cussen, L. D., Høghoj, P., & Anderson, I. S. (2001). Neutron collimator with rectangular beam profile. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *460*(2–3), 374–380. https://doi.org/10.1016/s0168-9002(00)01077-9

d'Errico, F., Giusti, V., Reginatto, M., & Wiegel, B. (2004). A telescope-design directional neutron spectrometer. *Radiation Protection Dosimetry*, *110*(1–4), 533–537. https://doi.org/10.1093/rpd/nch278

d'Errico, Francesco, Di Fulvio, A., Maryañski, M., Selici, S., & Torrigiani, M. (2008). Optical readout of superheated emulsions. *Radiation Measurements*, *43*(2–6), 432–436. https://doi.org/10.1016/j.radmeas.2008.02.011

Davis, A. C., Kornreich, D. E., & Lambert, M. J. (2011). *Pandemonium: Bringing Order to Dose Calculations in Complicated Glovebox Arrays, Version 2.0 (LA-UR-11-05816)*. LANL.

Fette, I., Google, Inc., Melnikov, A., & Isolde Ltd. (2011). The WebSocket Protocol. Retrieved from Ietf.org website: https://tools.ietf.org/html/rfc6455

FLIR. (2019). Grasshopper3 USB3 | FLIR Systems. Retrieved from Flir.com website: https://www.flir.com/products/grasshopper3-usb3/

Friedmann, M., & Rauch, H. (1970). Neutron focusing by a curved soller collimator system. *Nuclear Instruments and Methods*, *86*(1), 55–59. https://doi.org/10.1016/0029-554x(70)90035-2

Gelhaus, F. E., & Roman, H. T. (1990). Robot applications in nuclear power plants. Progress in Nuclear Energy, 23(1), 1–33. https://doi.org/10.1016/0149-1970(90)90012-t

Goodrich, M. A. (2004). Using models of cognition in HRI evaluation and design. *Proc. of the AAAI 2004 Fall Symposium Series.*

Goorley, T., et al. (2012). Initial MCNP6 Release Overview. *Nuclear Technology*, (180), 298–315.

Hansen, T. C., Henry, P. F., Fischer, H. E., Torregrossa, J., & Convert, P. (2008). The D20 instrument at the ILL: a versatile high-intensity two-axis neutron diffractometer. *Measurement Science and Technology*, *19*(3), 034001. https://doi.org/10.1088/0957-0233/19/3/034001

Imaida, I., Muraki, Y., Matsubara, Y., Masuda, K., Tsuchiya, H., Hoshida, T., … Ikeda, H. (1999). A new tracking satellite-borne solar neutron detector. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *421*(1–2), 99–112. https://doi.org/10.1016/s0168-9002(98)01163-2

John S. McCain National Defense Authorization Act for Fiscal Year 2019, H.R. 5515, 115th Cong., (2018).

KAERI. (n.d.). Nuclide Table. Retrieved from http://atom.kaeri.re.kr:8080/ton/index.html

Kawatsuma, S., Fukushima, M., & Okada, T. (2012). Emergency response by robots to Fukushima-Daiichi accident: summary and lessons learned. *Industrial Robot: An International Journal*, *39*(5), 428–435. https://doi.org/10.1108/01439911211249715

Knoll, G. F. (Ed.). (2000). *Radiation detection and measurement*. New York, NY: Wiley.

Kobayashi, T., Miyajima, K., & Yanagihara, S. (2002). Development of remote surveillance squads for information collection on nuclear accidents. *Advanced Robotics*, *16*(6), 497–500. https://doi.org/10.1163/156855302320535818

LANL. (2018). LANL - MCNP: Frequently Asked Questions. Retrieved from Lanl.gov website: https://laws.lanl.gov/vhosts/mcnp.lanl.gov/mcnp_faq.shtml

Los Alamos National Laboratory. (2005). *Plutonium pits are cast at LANL*. Retrieved from https://www.flickr.com/photos/losalamosnatlab/6876770217/in/album-72157629415211224/

Los Alamos National Laboratory. (n.d.). Our History. Retrieved September 2016, from http://www.lanl.gov/about/history-innovation/index.php

Los Alamos National Laboratory. (n.d.-b). TA-55 PF-4 LANL Plutonium-Processing Facilities [Pamphlet]. Retrieved from http://www.lanl.gov/about/_assets/docs/fact-sheets/ta-55-factsheet.pdf

Los Alamos National Laboratory, Operated by Los Alamos National Security, LLC, for the U.S. Department of Energy, 2015. Retrieved from https://www.lanl.gov/newsroom/picture-of-the-week/pic-week-12.php

Luszik-Bhadra, M., d'Errico, F., Hecker, O., & Matzke, M. (2002). A wide-range direction neutron spectrometer. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *476*(1–2), 291–297. https://doi.org/10.1016/s0168-9002(01)01442-5

Maimone, M., Matthies, L., Osborn, J., Rollins, E., Teza, J., & Thayer, S. (1998). A Photo-Realistic 3-D Mapping System for Extreme Nuclear Environments: Chornobyl. *Proc. of the 1998 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*.

Marder-Eppstein, E., Lu, D. V., & Hershberger, D. (2018). costmap_2d - ROS Wiki. Retrieved from Ros.org website: http://wiki.ros.org/costmap_2d

Martz, R. L. (2011). MCNP6 Unstructured Mesh Capability (U) (Technical Report No. LA-UR-11-02767).

Neutron Soller Collimator | JJ X-RAY. (2019). Retrieved from Jjxray.dk website: https://www.jjxray.dk/products/foil-collimators/neutron-soller-collimator

Optris. (2015). Spot finder IR camera Xi 400. Retrieved from Optris.com website: https://www.optris.com/optris-xi-400

Parasuraman, R., Sheridan, T. B., & Wickens, C. D. (2000). A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 30(3), 286–297. https://doi.org/10.1109/3468.844354

Park, S.-T. (2003). Neutron energy spectra of 252Cf, Am-Be source and of the D(d,n)3He reaction. Journal of Radioanalytical and Nuclear Chemistry, 256(1), 163–166. https://doi.org/10.1023/a:1023333016692

Pérot, B., Jallu, F., Passard, C., Gueton, O., Allinei, P.-G., Loubet, L., … Carrel, F. (2018). The characterization of radioactive waste: a critical review of techniques implemented or under development at CEA, France. *EPJ Nuclear Sciences & Technologies*, *4*. https://doi.org/10.1051/epjn/2017033

Peurrung, A. (2000). Recent developments in neutron detection. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *443*(2–3), 400–415. https://doi.org/10.1016/s0168-9002(99)01165-1

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., … Ng, A. Y. (2009). ROS: An Open-source Robot Operating System. *Proceedings of Proc. Open-Source Software Workshop of the International Conference on Robotics and Automation (ICRA)*.

Quinlan, S., & Khatib, O. (1993). Elastic Bands: Connecting Path Planning and Control. *Proc. IEEE International Conference on Robotics and Automation*, 802–807.

Ravishankar, R., Bhaumik, T. K., Bandyopadhyay, T., Purkait, M., Jena, S. C., Mishra, S. K., … Pal, P. K. (2013). Radiation mapping inside the bunkers of medium energy accelerators using a robotic carrier. *Applied Radiation and Isotopes*, *80*, 103–108. https://doi.org/10.1016/j.apradiso.2013.06.017

Reilly, D., Ensslin, N., Hastings Smith, Kreiner, S., Los Alamos National Laboratory (U.S., & Etats-Unis. Nuclear Regulatory Commission). (1991). Passive Nondestructive Assay of Nuclear Materials. Springfield, VA: Us Department Of Commerce, National Technical Information Service.

Reilly, J. K., McIntosh, T. W., Northey, L. M., GaTanto, J. J., Osterhoudt, T. R., & Thompson, J. D. (1985). Processing and Removal of the Three Mile Island Makeup and Purification System Resins. *Waste Management*.

Riley, V. (1989). A General Model of Mixed-Initiative Human-Machine Systems. *Proceedings of the Human Factors Society Annual Meeting,* 33(2), 124–128. https://doi.org/10.1177/154193128903300227

Rinard, P. (n.d.). Neutron Interactions with Matter. Retrieved from http://www.lanl.gov/orgs/n/n1/panda/00326407.pdf

Roman, H. T. (1991). Robots cut risks and costs in nuclear power plants. *IEEE Computer Applications in Power*, *4*(3), 11–15. https://doi.org/10.1109/67.85957

ROS.org. (2018). ROS/Introduction - ROS Wiki. Retrieved from Ros.org website: http://wiki.ros.org/ROS/Introduction

Ryan, J. M., Castaneda, C. M., Holslin, D., Macri, J. R., McConnell, M. L., Romero, J. L., & Wunderer, C. B. (1999). A scintillating plastic fiber tracking detector for neutron and proton imaging and spectroscopy. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *422*(1–3), 49–53. https://doi.org/10.1016/s0168-9002(98)01061-4

Scheer, N. L., Sanchez, C., Rebeil, J. P., Mitchel, E. A., & Yagow, C. (2002). *ARIES Robotic Integrated Packaging System User Guide (LA-UR-02-6855)*. LANL.

Sheridan, T. B., & Verplank, W. L. (1978). *Human and computer control of undersea teleoperators*. MIT Man-Machine Systems Lab.

Shultis, J. K., & Faw, R. E. (2010). Radiation Shielding and Radiological Protection. In D. G. Cacuci (Ed.), *Handbook of Nuclear Engineering*. Springer US.

Sofer, Z., Šimek, P., Jankovský, O., Sedmidubský, D., Beran, P., & Pumera, M. (2014). Neutron diffraction as a precise and reliable method for obtaining structural properties of bulk quantities of graphene. *Nanoscale*, *6*(21), 13082–13089. https://doi.org/10.1039/c4nr04644g

Standards for Protection Against Radiation, 10 C.F.R. § 20. (1991).

Taylor, G. C. (2010). Design of a direction-dependent neutron dosimeter. *Radiation Measurements*, *45*(10), 1301–1304. https://doi.org/10.1016/j.radmeas.2010.08.019

TEPCO. (2019). Area Around Reactor Buildings | TEPCO. Retrieved from Tepco.co.jp website: https://www7.tepco.co.jp/responsibility/decommissioning/robot/inbuildings-e.html

Tsitsimpelis, I., Taylor, C. J., Lennox, B., & Joyce, M. J. (2019). A review of ground-based robotic systems for the characterization of nuclear environments. *Progress in Nuclear Energy*, *111*, 109–124. https://doi.org/10.1016/j.pnucene.2018.10.023

Tsoulfanidis, N., & Landsberger, S. (2011). *Measurement and detection of radiation.* Boca Raton, FL: CRC Press.

Universal Robots. (2018). *Universal Robots e-Series User Manual Version 5.0.2*.

Universal Robots. (2019). UR5 collaborative robot arm | flexible and lightweight robot arm. Retrieved from Universal-robots.com website: https://www.universal-robots.com/products/ur5-robot/

U.S. Government Accountability Office. (2016, August). DOE Project Management: NNSA Needs to Clarify Requirements for Its Plutonium Analysis Project at Los Alamos (Report No. GAO-16-585). Retrieved from http://www.gao.gov/products/GAO-16-585

Vanier, P. E., Forman, L., Dioszegi, I., Salwen, C., & Ghosh, V. J. (2007). Calibration and testing of a large-area fast-neutron directional detector. *2007 IEEE Nuclear Science Symposium Conference Record*. https://doi.org/10.1109/nssmic.2007.4436312

Veirs, K. (2018, April 26). *Personal communication*.

von Frankenberg, F., McDougall, R., Nokleby, S., & Waller, E. (2012). A Mobile Robotic Platform for Generating Radiation Maps. *Intelligent Robotics and Applications*, 407–416. https://doi.org/10.1007/978-3-642-33515-0_41

Werner, C. J. (2017). *MCNP Users Manual - Code Version 6.2*. Los Alamos National Laboratory. Report LA-UR-17-29981